True BASIC 语言结构化程序设计

孙家启　周煦　吴国凤　编著

兵器工业出版社

True BASIC语言结构化程序设计

孙家启　周　煦　吴国凤　编著
王杰臣　张延炬　主审

兵器工业出版社

（京）新登字049号

内容简介

True BASIC语言是BASIC语言创始人 J·G 凯梅尼和 T·E 库尔茨对BASIC语言作了重大的改进和扩充后推出的新版本。本书系按照安徽省高等学校 True BASIC语言水平考试大纲和机械部高校计算机基础教育语言设置教学大纲要求编写而成。

全书共12章：第1章电子计算机概论，第2章磁盘操作和怎样使用True BASIC，第3章True BASIC的基本概念及基本语句，第4章分支结构，第5章循环结构，第6章数组与矩阵，第7章函数和子程序，第8章结构化程序设计，第9章字符处理，第10章格式化输入与输出，第11章文件，第12章图形。本书还附有True BASIC有关附录。

本书内容由浅入深、简明扼要，适于作为大专院校教材，也是计算机应用工作者提高语言能力的自学参考书。

True BASIC语言结构化程序设计
孙家启　周[illegible]

责任编辑

*

兵器工业出版社　出版发行
（北京市海淀区车道沟10号）
各地新华书店经销
通县建新印刷厂印装

*

开本　787×1092　1/16　印张：14.5　字数：356　千字
1993年8月第1版　1993年8月第1次印刷
印数：1--3500　定价：11.50元
ISBN　7-80038-646-5/TP·45

前言

从科学技术发展的角度，可以认为人类已经跨入信息和计算机时代。计算机的应用，已从科学家的实验室步入生产领域、交换领域、管理领域，直至家庭生活领域，从宏观的宇宙航天事业到微观的分子研究，几乎无处不有计算机的存在，并且它的发展正以等比级数的速度向前推进。普及计算机知识，全面推广计算机应用，不论人们主观意向如何，业已成为历史的必然趋势。

BASIC语言是程序设计语言中简单、易学、很受用户欢迎的语言。20世纪70年代以来，迅速发展的微型计算机，全部配有这种语言，使得BASIC成为世界上应用最广泛的一种计算机语言。但在利用内存空间、编辑功能以及实现程序结构化方面比不上PASCAL和C语言。BASIC创始人John G.Kemeny和Thomas E•Kurtz总结了BASIC语言使用经验，吸取了其他高级语言的优点，引入了结构化程序设计的思想，对原有的BASIC语言作了重大修改，于1985年11月公布了新版本的BASIC语言，并命名为“True BASIC”（真正的BASIC）。

True BASIC保留了原BASIC语言的通俗易懂，易学易用的突出优点，它严格遵循美国国家标准BASIC的规定，并对它作了重大改进和扩充。True BASIC已被美国推荐为国家标准版本，它是一种理想的完全结构化的程序设计语言，它可以和PASCAL语言相媲美，在某些方面甚至比PASCAL语言还好。它是一种增加了许多极强功能的语言，强化了全屏幕编辑功能和高效的图形功能等。另外，True BASIC提供了解释方式和编译方式，可以把编辑、编译、运行、追踪等工作融为一个整体，而且它具有很好的可移植性，能给I/O接口和汇编语言提供较好的访问方式和连接手段，开辟了比原BASIC大得多的用户空间，由64K扩大到512K或640K。总之，True BASIC确实是学习程序设计的理想语言。

本书内容由浅入深，循序渐近，突出重点，系统性强。全书共分十二章。第一章介绍了计算机基础知识和BASIC语言的发展以及True BASIC的特点；第二章介绍了磁盘操作系统和操作命令以及True BASIC语言上机操作的基础知识和汉化系统的应用；第三章介绍了True BASIC基本概念、基本语句和简单程序设计的基本知识；第四、第五、第六章分别介绍了分支结构 循环结构和数组与MAT语句，这部分内容是程序设计的重要基础，是本书的重要内容之一 强调和注重实际编程的基本训练；第七、第八章函数与子程序，结构化程序设计，也是本书的重点内容之一，主要介绍了内部函数、内部子程序、外部函数、外部子程序、库文件和程序 分解与链接等概念，强调结构化和模块化程序设计方法与技巧；第九章介绍了字符型变量的运算和字符型函数的应用；第十章格式化输入与输出，主要介绍了屏幕的光标定位控制和自选打印格式语句的显示应用；第十一章介绍了True BASIC的文件，其中包括顺序文件、记录文件和字节文件，它们各具特色，用途非常广泛；第十二章图形是本书的又一重点内容，True BASIC具有很强的图形功能，它的绘图语句简单、直观，同时，可以定义子程序形式的图画程序块，应用很广。

本书内容丰富完整，系统性强，兼顾初学与提高的层次，充分体现True BASIC最新版

本的完整与优越性能。语言通俗易懂，例题丰富，并附有丰富的应用开发指导性资料，不仅可作为非计算机专业的计算机水平考试教材，而且可供已有一些BASIC语言知识，要求提高程序设计与开发能力的应用工作者用作进修提高的读本。

本书以IBM PC微机及其兼容机为标准，True BASIC必须在磁盘操作系统DOS2·0或2·0以上版本支持下才能运行。

本书可根据不同专业、不同层次的要求，对教材内容适当取舍。课堂讲授40～60学时为宜。在教学过程中，必须注意和加强上机实践环节，以获得较好地效果。

本书第二、三、四、五章由合肥工业大学吴国凤同志编写；第一、六、九、章由合肥工业大学周煦同志编写，第七、八、十、十一、十二章由合肥工业大学孙家启同志编写。

哈尔滨电工学院王杰臣教授、北京机械工业学院张延炬副教授认真审阅了全书，并提出了宝贵的意见。在本书编写过程中，得到了机械电子工业部高教处郭莹和王清池两同志大力支持和帮助，在此一并表示感谢。由于编著者水平有限，书中难免存在错误和不当之处，诚恳希望读者批评指正。

编　者

1993年1月

目录

第一章 电子计算机概论…………（1）

§ 1-1 电子计算机的特点和用途……（1）

§ 1-2 电子计算机系统的组成………（4）

§ 1-3 数据在计算机内的存储方式…（8）

§ 1-4 BASIC语言的发展和True BASIC的特点………………（13）

§ 1-5 利用计算机解决实际问题的步骤（14）

习题一…………………………（14）

第二章 磁盘操作系统和怎样使用 True BASIC……………（16）

§ 2-1 磁盘操作系统的基本功能和启动…………………………（16）

§ 2-2 PC-DOS操作系统的基本命令…（20）

§ 2-3 汉字操作系统的功能与应用…（27）

§ 2-4 怎样使用True BASIC…………（29）

习题二…………………………（34）

第三章 True BASIC 的基本概念及基本语句…………………（36）

§ 3-1 基本概念………………………（36）

§ 3-2 基本语句………………………（42）

§ 3-3 串变量在提供数据语句中的使用（49）

§ 3-4 几个简单语句…………………（52）

§ 3-5 应用举例………………………（54）

习题三…………………………（55）

第四章 分支结构………………（57）

§ 4-1 概述……………………………（57）

§ 4-2 程序流程图……………………（57）

§ 4-3 IF-THEN分支控制结构………（58）

§ 4-4 情况选择分支控制结构………（62）

§ 4-5 应用举例………………………（64）

习题四…………………………（67）

第五章 循环结构………………（69）

§ 5-1 循环的概念……………………（69）

§ 5-2 FOR-NEXT 循环结构…………（70）

§ 5-3 DO循环控制结构………………（74）

§ 5-4 应用举例………………………（80）

习题五…………………………（83）

第六章 数组与矩阵……………（85）

§ 6-1 数组的基本概念………………（85）

§ 6-2 一维数组及其应用……………（88）

§ 6-3 二维数组及其应用……………（94）

§ 6-4 MAT语句………………………（97）

习题六…………………………（106）

第七章 函数和子程序…………（109）

§ 7-1 标准函数………………………（109）

§ 7-2 自定义函数……………………（118）

§ 7-3 子程序…………………………（130）

习题七…………………………（138）

第八章 结构化程序设计………（140）

§ 8-1 整体变量和局部变量…………（140）

§ 8-2 外部模块………………………（142）

§ 8-3 程序的分解与链接……………（153）

习题八…………………………（158）

第九章 字符处理………………（159）

§ 9-1 字符型变量的运算……………（159）

§ 9-2 字符串函数的应用……………（161）

习题九…………………………（166）

第十章 格式化输入与输出……（167）

§ 10-1 屏幕的光标定位控制………（167）

§ 10-2 自选打印格式语句…………（170）

习题十…………………………（175）

第十一章 文件…………………（176）

§ 11-1 文件的基本概念……………（176）

§ 11-2 顺序文件（文本文件）………（177）

§ 11-3 记录文件……………………（186）

§ 11-4 字节文件……………………（192）

习题十一………………………（194）

第十二章 图形…………………（195）

§ 12-1 作图环境……………………（195）

§ 12-2 标准作图语句………………（198）

§ 12-3 动画与图画…………………（203）

§ 12-4 图形输入与多窗口操作……（207）

§ 12-5 统计图的绘制………………（210）

习题十二………………………（212）

附录 ……………………………… （213）
附录一　ASCII码表 …………………… （213）
附录二　True BASIC 语法总结 ……… （214）
附录三　从BASIC到True BASIC的转换软件…………………………… （221）
附录四　True BASIC与DOS操作系统间的接口方法…………………… （224）
附录五　True BASIC与其他可执行程序（·EXE文件）间的接口方法………………………………… （225）
参考文献……………………………… （226）

第一章 电子计算机概论

电子计算机（Electronic Computer）是一种具有存储能力、逻辑判断与推理能力，由程序控制以高速度和高精度自动进行运算和操作的电子装置。俗称电脑。它的出现和发展，有力地推动着科学技术的迅速发展，促进现代化管理水平的提高，使整个社会发生着极其深刻的变化，把人类带入了信息时代，是本世纪科学技术最卓越的成就之一。

本章将粗略地介绍一些必要的计算机基础知识，深入浅出地介绍计算机的基本工作原理以及数据在计算机内的存储方式和计算机解题的方法和步骤，使读者对计算机系统有一个整体的了解，并为学好True BASIC语言打下基础。

§1-1 电子计算机的特点和用途

一、电子计算机的特点

电子计算机是现代人们掌握的最先进的工具之一，它主要有以下几个方面的特点：

1. 运算速度快

能进行快速运算是电子计算机最显著的特点。这是由于计算机采用了高速的电子器件和线路，并利用先进的计算技术，因此使得计算机可以有很高的运算速度。一般的计算机运算速度每秒可达几十万次到几百万次，有的可达几千万次到几亿次。现在有些高档计算机的运算速度甚至可达几十亿次到几百亿次以上。例如，伟大的数学家契依列花了15年的时间，将圆周率π计算到707位，而现在用一般的计算机8小时就可将π算到1万位，这样的高速运算人工是无法比拟的。

2. 精确度高

计算机的精确度取决于字长，字长越长，精确越高。一般微型机字长为4～16位，也有32位的，小型机字长为16～32位，中型机字长多为32位，巨型机或大型机字长一般为32～64位。通常计算机能进行双倍字长或多倍字长运算。例如在IBM-PC机上，作9/17的除法，可达到16位有效数字，即9/17＝0.5294117647058824。这样高的计算精度，一般计算工具是很难达到的。

3. 具有计忆能力

计算机中的存储器（内存和外存）能够存储大量信息。它能把数据、程序存入进行处理，计算并把结果保存起来，这是电子计算机区别其他计算工具的较本质的特点。例如，一般计算器至多只能存放少量数据，而电子计算机却能存储几万，几十万乃至几十亿个数据；一般计算器不能存放计算程序，而电子计算机能将程序存放起来，当运行时，能高速地从原来存放的地方依次取出，逐一加以执行，这样，不要人去干预就能自动地完成运算。

4. 具有逻辑判断能力

计算机不仅能进行算术运算，还具有逻辑判断功能。例如，对两个信息进行比较，计算

机根据比较的结果，会自动地确定下一步该做什么，遇到分支能按条件选择支路。正因为计算机具有了逻辑判断能力，使得计算机的应用，进入了资料分析、情报检索、逻辑推理、过程控制、数据处理、定理证明等工作领域，大大扩大了计算机的应用范围。

5. 自动化程度高

计算机从正式开始工作，到送出结果，整个工作过程都是在程序的控制下自动进行的，完全用不着人去参与，这也是计算机最突出的特点。自动化程度高低，主要取决于两个因素：①具有“记忆”能力；②具有逻辑判断和选择能力。

二、电子计算机的用途

电子计算机的用途非常广泛，许多新的领域正在不断开辟。正如有人作过这样的描述：显微镜、望远镜和雷达是人眼睛功能的延长；各种机床、机械工具是人手功能的延长；而电子计算机则是人大脑功能的延长。

归纳起来，电子计算机的应用范围主要有以下几个方面：

1. 数值计算

电子计算机能高速度、高精度地完成各种数值计算，被广泛地应用在科学技术领域中。随着计算技术的不断发展，电子计算机和许多基础科学相结合，出现了一系列新兴的边缘科学，象计算数学、计算物理学、计算天文学、计算地质学、计算生物学、计算力学等。可以预计，将来还会不断出现一些新的计算科学领域。

过去，很多工程设计和科研课题，由于计算工作量的庞大而无法进行，或只能用粗略的近似计算。但采用计算机后，由于它具有快速、精确的特点，过去人工计算需要几个月，或几年时间才能完成的工作量，现仅要几天，或几个小时，甚至几分钟就能解决了。例如，天气预报，天气的变化是由地球表面大气的运动造成的，这种大气的运动可以用一组微分方程式描述，求解这组方程的计算量是庞大的。40多年前，用人工计算一个地区3小时后的气象变化，要用6万多人计算，才能赶上天气的变化。要不然，天气预报就成“马后炮”了。现在，用一般水平的电子计算机，算一个地区的4天的天气形势预报，只需十几分钟。

而且使用了电子计算机，能使科学实验时间大量减少，得到很好的经济效益。据说，第二次世界大战中，德国的V-2火箭实验发射了1400次之多，结果证明对英国伦敦的袭击命中率非常低。而现代的火箭，只要发射几次，就可以定型。这是因为事先作了大量的计算。

2. 数据处理和信息加工

它能存储大量的数据，并能快速地处理。这在信息处理、企业管理、情报检索及信息传递等方面有着广泛的应用。例如，美国的医学图书馆的自动检索系统，可以在10分钟内，用30种语言浏览2300种杂志的十万篇文章，查阅完一个指定的课题。这样的工作量，如果由人来干，即使不计及语言的知识，也至少要五年的时间。

现在，数据和信息的处理在计算机的使用中占有很大的比重，随着社会的不断发展，比重将越来越大。一些先进国家中的大企业、政府部门、机关学校都使用大型信息和数据处理系统进行高效的管理。微机的出现促使现代化电脑管理进入以文献信息为主的多种事务管理领域。尤其在办公自动化领域中，微机已大显身手。对于我国使用汉字为主的国家，在汉字处理方面，我国的科技人员经过多年的努力，已研制成多种类型的汉字信息处理系统，这对我国来说，是实现现代化电脑管理的关键。

3. 自动控制

利用计算机和其它自动控制设备组成计算机自动控制系制，可实现对工业过程的自动控制，它能代替人们对某些工农业生产过程进行监视和控制，提高产品的质量、减轻劳动强度、提高劳动生产率。例如，数控线切割机床系统、锅炉微机自动控制系统，纺织过程微机控制系统，化工配比微机控制系统，交通信号控制系统等等。

有一些控制问题，是人们无法去亲自操作的。例如，宇宙飞行、火星探测等，就要用电子计算机精确地控制。再如，对现代军用飞机的控制，要求在很短的时间内，计算出敌机的各种飞行参数，以便控制自己飞行的姿态，确定采用什么样的攻击方案，决定利用何种武器。这些控制，对于驾驶员来说，是很难承受的负担。所以，现代飞机把计算机作为控制中心，据说B-1飞机上就有计算机近30台。

近年来，由于微处理机和单片机的出现，进一步扩大了计算机在自动控制方面的应用范围。

4. 计算机辅助设计

计算机辅助设计，简称CAD技术，是综合地利用电子计算机的工程计算、逻辑判断、数据处理功能和人的经验与判断能力结合，形成一个专门系统，用来帮助人们为许多工业部门快速、高质量地设计复杂的优质产品，并能加速现代工业产品的更新换代。它是近十多年来形成的一个重要的计算机应用领域。目前在飞机、船舶、集成电路、大型自动系统等的设计中，CAD技术有愈来愈重要的地位。

利用CAD技术来设计和制造计算机，就是一个突出的例子。随着计算机本身的功能越来越多，设计工作量也显著增加。一台计算机的设计，从方案论证、逻辑设计、组装设计到制造、调试，是个复杂而烦琐的过程。采用CAD技术，大部分设计工作，包括各种工程逻辑图，插件布线图的绘制，都可以利用计算机进行。在计算机的生产过程中，经常利用计算机协助筛选器件和测试插件，甚至整机的测试和诊断，使得生产周期大为缩短，可靠性大大提高。

除此之外，现在工业中还实现了计算机辅助测试（CAT）、计算机辅助制造（CAM）和计算机辅助工程（CAE）。

5. 人工智能

人工智能是指计算机模拟人脑的部分功能，学习与思考、推理与判断、作文字翻译，用普通语言与人对话等等。第五代计算机就是以智能计算机为代表。它是计算机科学中的一个重要分支。

人工智能活动是一种高度复杂的脑力功能，如人的脑功能有联想记忆、模式识别、语言翻译、学习模仿、归纳演绎、决策对奕、文艺创作、创造发明等，都是一种高度复杂的生理和心理活动过程。总之，人的智能有极其复杂的数学模型，而且这种模型在短期内很难找到，因此，当前所能做到的，仅是利用计算机系统协助人们作一些在特殊情况下的智能方面的工作，并逐步扩大应用范围。

一个专家系统就是具有大量专门知识的程序系统，能处理和解决这一领域内的专门问题。例如，中医诊治系统，能作600多种不同的数学运算的MACSYMA模式制导系统，DENDRAL的化学质谱分析系统等。

§1-2　电子计算机系统的组成

一个完整的计算机系统由硬件和软件两部分组成。目前一个计算机系统是否成功，软件丰富与否，这是至关重要的。成功的计算机系统，可选软件相当多，软件的成本超过硬件的成本。仅仅有计算机硬件就谈不上计算机的应用。同样的计算机硬件，使用低劣的计算机软件和使用优秀的计算机软件，将会得到不同的名声。往往许多低水平的使用，会使计算机的声誉不高，阻碍计算机的推广。现在许多同志购买计算机系统时，往往把眼睛盯到硬件上，而忽视了软件，加之使用人员素质低，造成某些系统闲置不用，或利用率很低。因此，我们千万不能只重视计算机硬件而忽视软件。

组成计算机的物质设备，我们称它为计算机硬件。它是可以看得见、摸得着的电子机械装置。

为了充分发挥计算机的效能，用户所使用的各种程序系统及其相应的文件，被称之软件。

重程序而轻文件的倾向，社会上是有的，但是没有相应的文件，多数有实用价值的程序也是不能产生的，即使产生了，要使用一个程序也是困难的，维护它更是难以办到的。因此，任何国家的软件登记都规定要文件齐全，而不能仅仅是程序。

一、电子计算机的硬件

运算器、控制器、存储器、输入设备、输出设备等构成了计算机硬件，见图1-1所示。

1. 存储器

存储器是计算机的记忆装置，它是存储信息（程序和数据）的部件，其作用如同录音机的磁带，能把信息保存起来，也可以通过新的信息把原来的信息抹去（冲掉）。计算机的存储器由一个个的存储单元组成，每一个存储单元有一个编号，用以区分不同的单元，通常把这个编号称做存储单元的地址。程序和数据分别被存放在这些单元中，见图1-2所示。计算机工作时，根据地址找到对应的存储单元，进行信息的存、取操作。

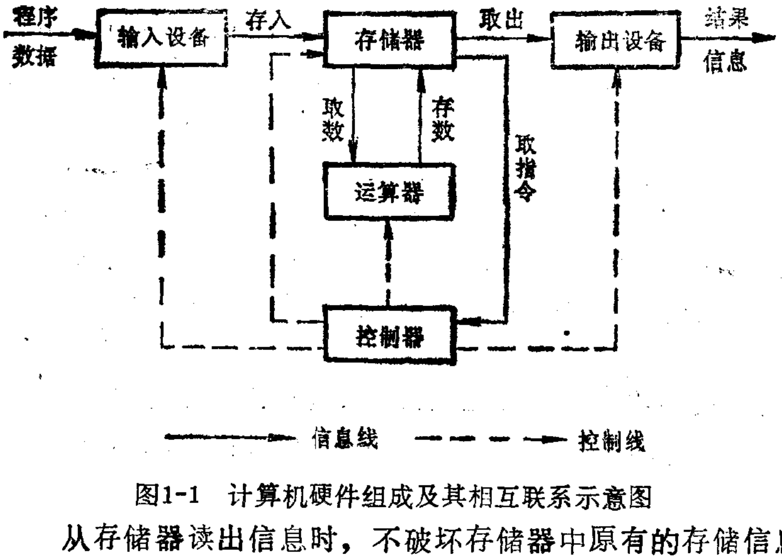

图1-1　计算机硬件组成及其相互联系示意图

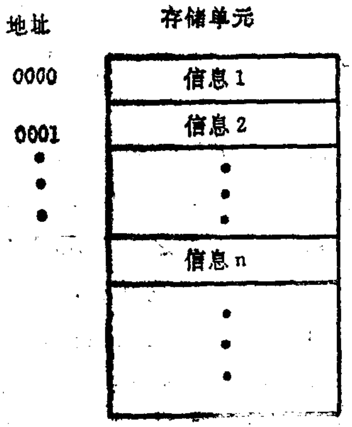

图1-2　存储器示意图

从存储器读出信息时，不破坏存储器中原有的存储信息，往存储器写入信息时，新信息将代替原有的存储信息，原有的存储信息将丢失。故存储器称之为：“取之不尽，新来旧

去。* 这是存储器的一个重要特点。

存储器的容量是存储器的指标之一，常常用××k表示。比如64k、128k、256k、512k等。1k等于1024，这里k的含义，一般是指“千字节”，即kB，即1k为1024个“字节”(byte)。对于大容量的存储器，常用××兆，一兆等于1024k，即1MB。

2.运算器

运算器是用来快速地进行各种算术运算和逻辑运算的部件。它由一系列小存储部件（称为寄存器）、加法器等能作算术操作和逻辑操作的电子器件、线路组成。必须指出，运算器只能完成简单的基本运算，对于复杂的问题，在计算机中只能用简单的方法一步一步实现。不过，计算机的运算速度快的惊人，它的威力和奇迹是靠速度赢得的。

字（word）和字长：作为一个整体来处理或运算的数据称为字。一个字所包含的位数称为字长。一般一个字的字长是字节的整数倍。

3. 控制器

控制器是计算机的指挥中心，用来控制计算机各部件协调一致自动地进行操作。概括地讲，它的工作是：取指令——从存储器中按序取出要执行的指令；分析指令——分析指令规定的操作内容；执行指令——根据对指令的分析，向有关部件发出控制信号，让它们执行规定的任务。

4.输入设备

程序及原始数据，通过输入设备转换成计算机能够识别的代码，并送到存储器里保存起来。比如键盘、纸带输入机、卡片阅读机、语音输入装置、图形数字化仪、光学字符识别设备、数据站等都是输入设备。

5. 输出设备

计算结果或人们所需要的其它信息，通过输出设备从计算机中传送出来。比如显示器、打印输出设备、数控绘图机、纸带穿孔机、卡片穿孔机、语音输出装置等都是输出设备。

二、电子计算机的软件

计算机软件包括系统软件和应用软件。系统软件包括操作系统、程序设计语言、非程序语言和实用程序。

1. 操作系统

操作系统是为管理计算机的硬件资源和软件资源而编制的程序，用户只能通过操作系统来使用计算机。其它软件只能在某一个操作系统上运行。经常在微机上运行的著名操作系统有DOS、CP/M、UNIX、UCSD、OASIS及它们的变种。

2. 程序设计语言

何谓语言？在社会科学中人们把语言称为是人与人之间交流思想的工具，而在自然科学中，语言则为一个字符表（集）上的句子的任何集合，字符表是字（母）、数字、符号的任何有穷集合。

句子是由字（母）、数字、符号组成的有限长度的集合。严格讲，计算机所使用的各类语言都是由一个有限的字符集，并由字符集构成语句与数据，按一定语法规则组成的任何集合。

程序设计语言（或称计算机语言）大致分为三类、机器语言、汇编语言和高级语言。

(1) 机器语言（Mechine Language)

机器语言由二进制代码组成，是计算机唯一可以直接执行的语言。用机器语言编写的程

序叫做机器语言程序或手编程序。任何其它语言都要转换成机器语言，计算机才能执行。程序由一系列“指令”组成。

一条机器指令用来控制计算机进行一种操作。

例如：00111110　00010100　表示把20送给寄存器A中

11000110　00100011　表示把35与与寄存器A中的数相加，其结果再送入寄存器A中

11010011　00101111　表示把寄存器A中的数送入显示器显示出来

01110110　停机

上述是一个简单的机器语言编写的程序，每条机器指令都是由0和1两种符号排列组成的。指令不同，0、1的排列次序也不同。由此程序可以看出机器语言可读性太差，编程太困难，不通用，因此目前极少使用机器语言直接编程。

(2) 汇编语言 (Assembly Language)

为了克服机器语言读写困难，人们创造了汇编语言。汇编语言是用一些符号代替机器语言的二进制编码，它与机器语言的命令是一一对应的。

例如：上述机器语言程序写成汇编语言程序为：

```
LD A, 20
ADD A, 35
OUT (47), A
HALT
```

汇编语言的不同指令是用不同的助记符来区别的。如：ADD表示加法指令，OUT表示输出指令，指令中使用的操作数也是采用助记符号或十进制数来表示的。如，程序中的A，20，35都是操作数。

从上述汇编语言程序来看，在可读性上与机器语言相比，有了一定的改进，但仍然是难以使用的语言。由于它是面向机器的语言，能最充分地发挥计算机硬件的性能，因此使用最频繁的程序、追求运行效率的程序常用汇编语言来编程。如操作系统多数用汇编语言写成。编写系统软件，这往往是计算机专业人员的事。再就是在自动控制应用中，程序不太长，却对程序的运行速度有极高的要求，也利用汇编语言来编程。也有时为了充分发挥计算机的性能，整个大问题的若干个小问题用汇编语言来编程，然后用高级语言来调用。若用汇编语言来编程起码对计算机硬件要有一定的了解，进一步应对所使用的操作系统的内部功能(而不是操作系统的使用) 有所了解。

用汇编语言编写的符号程序叫汇编语言源程序，计算机不能直接运行这种源程序，必须把它转换成机器语言程序，能由机器执行。汇编语言程序的运行过程，见图1-3。

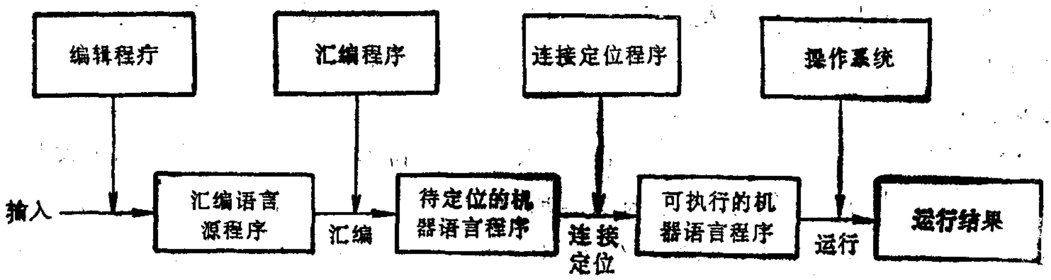

图1-3　汇编语言程序运行过程

从图1-3可以见到：首先通过实用软件编辑程序把编写好的汇编语言源程序送入计算机。然后再运行系统软件汇编程序，汇编程序把汇编语言源程序当成它的数据进行加工，加工的结果是待定位的机器语言程序。接下来，再运行实用软件连接定位程序，把待定位的机器语言程序转变成可执行的机器语言目标程序。这时通过操作系统运行可执行的机器语言目标程序，就可得出运算结果。

(3) 高级语言 (High Language)

高级语言是某种接近于人们习惯用的自然语言和数学语言的计算机程序设计语言，对于一个问题将它告诉计算机通过哪些步骤去解决，因此它是面向过程的语言。

由于客观世界问题的不同、程序员偏爱的不同，则使得高级程序设计语言发展了几百种之多。常见的有：BASIC、True BASIC、FORTRAN、COBOL、ALGOL、PASCAL、C、Ada、APL、FORTH、PL/1等高级语言。

你可能会想到，若只有一种高级语言进行所有的程序设计岂不更方便吗？实际上，由于计算机可以执行的任务太多，用某一种语言很容易执行的任务，而用另一种语言来执行则不一定容易，因此至今尚没有一种万能的计算机高级语言。目前高水平的应用，只能是对于大型题目使用多种语言解决，不同的子问题使用不同的语言，通过操作系统和实用程序，将各语言程序联在一起。这对程序员将有较高的要求。True BASIC语言是90年代设计的一种简单的高级语言，它仅含有当时所有高级语言的大部分基本特征。它是目前普及比较好的一种语言，用它开发的软件也很多。

例如：上例用True BASIC语言编写。

```
LET A=20
LET A=A+35
PRINT A
END
```

高级语言是独立于机器的，在编程序时人们不需要对机器的指令系统有深入了解，而且用高级语言编写的程序可以在不同型号的机器上使用，每条语句非常直观、功能相当强，可以直接引入和使用数学上的常数、变量、函数以及各种运算，这对于编写解决数学计算问题的程序非常方便。

但任何机器都不能直接执行高级语言，需要先处理一下，处理的办法有两类，一类是解释执行，另一类是编译执行，见图1-4。

解释执行的过程是，把用高级语言编的源程序输入计算机之后，有一个高级语言的解释程序（它是机器语言程序），把高级语言源程序逐句地翻译，翻译一句，执行一句，即边解释边执行。这种方式的最大好处是，对于程序中的错误，计算机会立即指出，因此对于初学者来讲，特别容易学习。缺点是运行速度低，因为每次执行都要执行它的解释程序。True BASIC语言可采用解释执行方式（也可采用编译执行方式）。True BASIC语言将要成为一个普及型的语种，在某种程度上与此有关。

编译执行的过程是，把高级语言源程序用编辑程序送入机器后，再执行编译程序。编译程序是把高级语言源程序当成数据进行加工，加工的结果是机器指令程序，编译过程中发现的源程序中的错误，必须重新用编辑程序修改，编译通过的程序，再执行定位连接程序，最终形成可执行的机器指令目标程序，最后通过操作系统来执行这个目标程序。这个目标程序

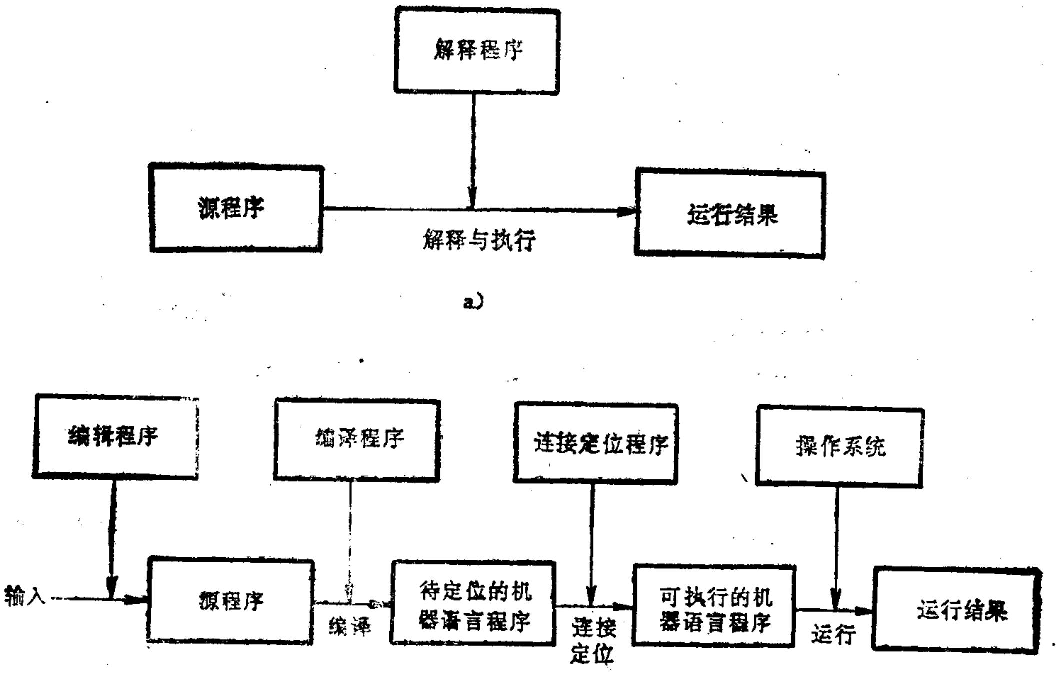
解释程序
源程序
解释与执行
运行结果
a)
编辑程序
编译程序
连接定位程序
操作系统
输入
源程序
编译
待定位的机器语言程序
连接定位
可执行的机器语言程序
运行
运行结果

图1-4 高级语言的两种执行过程
a)解释执行的过程 b)编译执行的过程

一旦形成，以后再执行就不再需要编辑程序、编译程序和连接程序了，连开始输入的高级语言源程序也不再需要了，只要这个目标程序就可以了。因此，它的优点是执行速度快，缺点是由于对整个程序的错误一起进行检查，因此对初学者来讲往往感到困难比较多。

3. 非程序设计语言

除了程序设计语言外，用于计算机的还有一种称为非程序设计语言的语言。也可称为面向问题的语言。它在使用时不必关心“怎么做”，而只说明“做什么”即可。

国外对这种易学、易用的非程序语言的评价较高，认为它是继硬件总线、软件总线之后发展起来的“应用总线”，是开拓微机或个人计算机市场的一大变革。

4. 实用程序

实用程序包括诊断程序、编辑程序、调试程序、打印程序、通讯程序、转换程序、连接程序和装入程序等。

5. 应用软件

应用软件的范畴非常庞杂，计算机应用就是通过应用软件来实现的，多数的程序人员都是在利用系统软件来开发应用软件。这里就不再一一列举。

整个电子数字计算机系统的组成，见图1-5。

§1-3 数据在计算机内的存储方式

一、内存的组织形式

电子计算机内的存储器（称内存储器，简称内存）是由千千万万个小的电子线路单元组成的，每一个单元有两种稳定的工作状态(例如三极管的截止和导通)，它们分别表代0和1。因此电子计算机存储的信息是用二进制形式存储的。每一个小元件中存放一个0或1的信

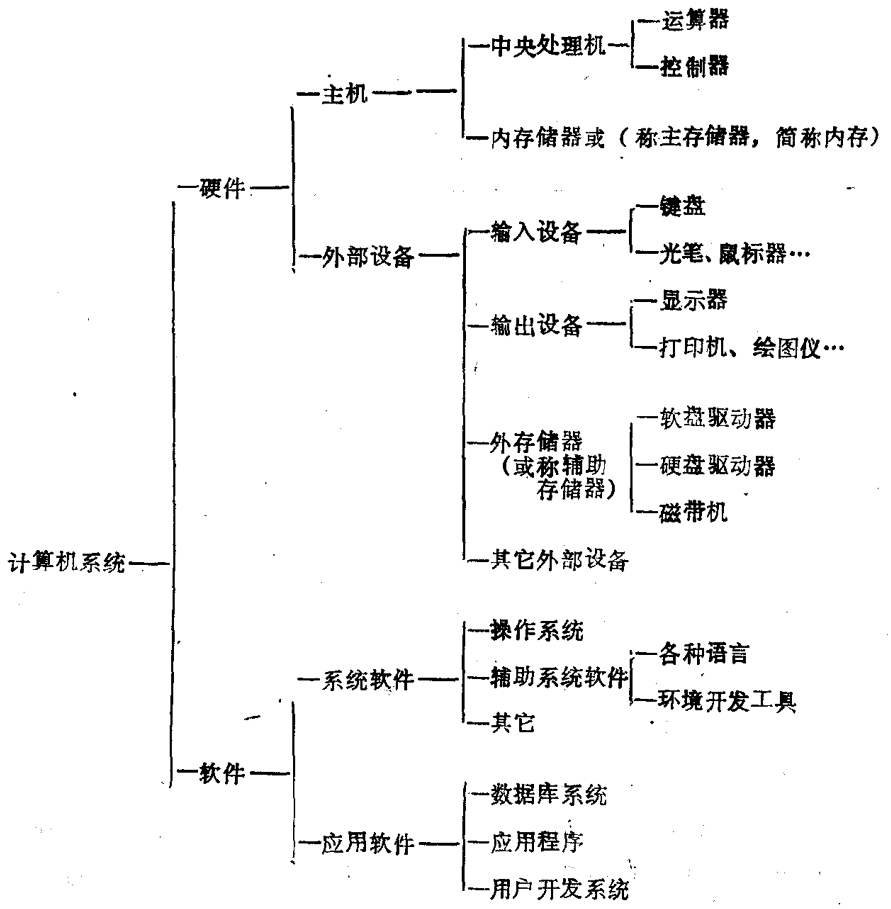
计算机系统
硬件
主机
中央处理机
运算器
控制器
内存储器或（称主存储器，简称内存）
外部设备
输入设备
键盘
光笔、鼠标器…
输出设备
显示器
打印机、绘图仪…
外存储器（或称辅助存储器）
软盘驱动器
硬盘驱动器
磁带机
其它外部设备
软件
系统软件
操作系统
辅助系统软件
各种语言
环境开发工具
其它
应用软件
数据库系统
应用程序
用户开发系统

图1-5　电子计算机系统的组成

息。关于内存，常用到以下一些术语：

位，又称比特（bit）。每一个能代表0和1的电子线路称为一个二进制位。一个存储器就是一个包含许许多多个二进位的电子单元的庞大电路。目前的存储器多采用集成化技术，把几万或几十万个电子元件做成一个集成电路，体积小，耗能少，速度快。

字节，又称拜特，（byte）。由若干个二进制位组成一个字节，多数计算机以8个二进制位组成一个字节。

字（word）。由若干个字节组成一个存储单元，称为“字”。一个存储单元中存放一条指令或一个数据。如果一个计算机以32个二进制的信息表示一条指令，就称这台计算机的“字长”为32位。

地址。为便于管理，对每个存储单元编一个存储单元号，这就是“地址”。通过地址可以找到所需的存储单元，可以取出其中存储的信息（或向指定的存储单元存入信息）。这如同旅馆中有许多房间，给每个房间编一个房号，只有通过房间号才能找到所要找的客人。请不要将存储单元的“地址”和存储单元中的“内容”这两个不同的概念混淆。正如同房间号和住在其中的旅客是两回事一样。见图1-6。

假设在地址为008的存储单元中放一个数5，在地址为016的存储单元中放数30，分别从这两个存储单元中将数取到运算器进行相乘，然后将乘积150送回地址为024的存储单元。

存储器、存储单元、字节、位之间的关系，见图1-7。

001	002	003	004	005	006
007	008　5	009	010	011	012
013	014	015	016　30	017	018
019	020	021	022	023	024　150

图 1-6

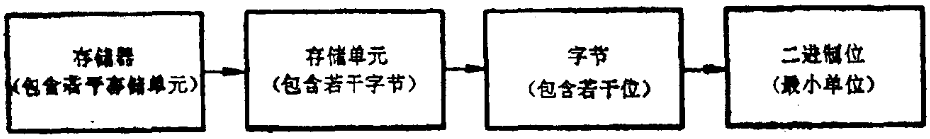

图1-7　存储器、存储单元、字节、位的关系

二、数据的二进制表示

人们最常用的数制是十进制，它是由0，1，2，3，4，5，6，7，8，9等十个不同的符号来表示数值的，这十个符号就是数字。在日常生活中，除了常用的十进制计数以外，也有不少非十进制的计数方法，如计时，60秒为1分，60分为1小时，是60进制计数法。中国的“老秤”，一斤等于16两，是十六进制计数法。一英尺等于12英寸，是十二进制计数法。当然，还有其他各种各样的进制计数法。

在计算机中，广泛使用二进制计数法，一个十进制数必须先化成二进制形式才能存放到计算机的存储单元之中，二进制的基本法则是：“逢二进一”。人们生活中经常遇到的鞋、袜、筷子、手套等就是逢二进一的实例。计算机之所以采用二进制只是因为许多电子元件具有两种稳定的工作状态（代表0和1）容易实现。

表1-1表示10以内的十进制数与二进制数的对照。

表　1-1

十进制数	化为以2为底的指数形式	二进制数
0	0×2^0	0
1	1×2^0	1
2	$1\times2^1+0\times2^0$	10
3	$1\times2^1+1\times2^0$	11
4	$1\times2^2+0\times2^1+0\times2^0$	100
5	$1\times2^2+1\times2^1+1\times2^0$	101
6	$1\times2^2+1\times2^1+0\times2^0$	110
7	$1\times2^2+1\times2^1+1\times2^0$	111
8	$1\times2^3+0\times2^2+0\times2^1+0\times2^0$	1000
9	$1\times2^3+0\times2^2+0\times2^1+1\times2^0$	1001

一个十进制数F可表示为：

$$(F)_{10}=a_0\cdot2^n+a_1\cdot2^{n-1}+\cdots+a_{n-1}\cdot2^1+a_n\cdot2^0$$

则a_0、a_1、a_2、a_3、…a_n就是F在二进制中的表示形式。欲求出a_0、a_1、…a_n，只需将F

不断被2除，取其余数即可。例如，求十进制数23的二进制形式如下：

$$\begin{array}{r|l}
2 & \underline{23}\ \ (1 \longleftarrow a_4 \\
2 & \underline{11}\ \ (1 \longleftarrow a_3 \\
2 & \underline{5}\ \ (1 \longleftarrow a_2 \\
2 & \underline{2}\ \ (0 \longleftarrow a_1 \\
2 & \underline{1}\ \ (1 \longleftarrow a_0 \\
 & 0
\end{array}$$

最后一个余数为a_0，从下向上依次为$a_0a_1a_2\cdots a_n$，因此 $(23)_{10}=(10111)_2$ 。括弧外的注脚“10”和“2”分别表示括弧中的数是十进制数和二进制数。

反之，二进制数可化成十进制数，见表1-2所示。

表 1-2

二进制数	十进制数
1	$2^0=1$
10	$2^1=2$
100	$2^2=4$
1000	$2^3=8$
10000	$2^4=16$
100000	$2^5=32$
1000000	$2^6=64$
10000000	$2^7=128$
⋮	⋮
$\underbrace{100\cdots\cdots000}_{n\text{个}0}$	2_n

如果一个二进制整数要化为十进制数，只要将其整数部分的最后一位乘以2^0，倒数第二位乘以2^1，……依此类推，小数部分则分别乘以2^{-2}……，将各项相加就得到用十进制数表示的数。

例如，二进制数10101101.1011所表示的值是：

$$\begin{aligned}
&1\times2^7+0\times2^6+1\times2^5+0\times2^4+1\times2^3+1\times2^2+0\times2^1+1\times2^0\\
&+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}+1\times2^{-4}\\
&=128+32+8+4+1+0.5+0.125+0.0625=173.6875
\end{aligned}$$

三、八进制和十六进制

由于二进制数位数多，数字冗长，不便记忆和书写，因此，常将二进制数分成三位一组，如110111010101可分为110，111，010，101四组，每一组代表一个0到7之间的数（三位数最大为111，相当十进制的7，即逢八进一）。因此，它是八进制数，即“逢八进一”。

$$\underbrace{110}_{6}\ \ \underbrace{111}_{7}\ \ \underbrace{010}_{2}\ \ \underbrace{101}_{5}$$

$(110111010101)_2=(6725)_8$，注意不是十进制的3725，而是八进制的6725。

如果将二进制数分成四位一组，就是十六进制数（四位的二进制数最大为1111，相当于十

进制的15，即逢十六进一）。如：

$$\underbrace{1001}_{9}\quad\underbrace{1101}_{D}\quad\underbrace{0101}_{5}$$

注意，在十六进制中把10以上的数用A（代表10）、B（代表11）、C（代表12）、D（代表13）、E（代表14）、F（代表15）表示。

表1-3举例说明十进制数和二进制数，八进制数、十六进制的对应关系。

表 1-3

十进制数	二进制数	八进制数	十六进制数
32	110000	40	20
62	111110	76	3E
114	1110010	162	72
241	11110001	361	F1
200	11001000	310	C8

在应用时，必须弄清楚所接触的数是什么进制的，不要弄错。例如对二进制中的100应读作“壹零零”而不应该读为“一百”。

以上只作简单的介绍，以便使读者对二进制数有一初步概念。实际上，在数据输入计算机时，是由计算机系统自动将十进制数转换为二进制数的，使用者不必自己转换，在输出时，由计算机将内存中的二进制数先转换成十进制数，然后输出给用户。初学者对八进制和十六进制可不必深究，知道有这一回事就行，以便今后查阅资料时不致茫然。

四、计算机中的编码

由于在计算机内部是以二进制数的形式进行存储、运算、识别和处理的，因此，在计算机中表示的数、字母和各种符号都要以特定的二进制编码来表示。目前国际上最普遍采用的是ASCII码，即美国标准信息交换码(American Standard Code for Information Interchange）。此外，在汉字处理中已开始使用我国自己创立的汉字编码。

1.ASCII码

ASCII码的编码表见附录1。它用七位二进制编码，故可表示128个字符，其中包括数字0～9及英文字母等可打印字符。在微型计算机中通常以8位二进制作为一个存储单元，故规定8位二进制为一个字节。所以，可用一个字节表示一个ACSII字符。例如，0～9的ASCII码为30H～39H，大写字母A～Z的ASCII码为41H～5AH。(字母H表示十六进制，不是数)。

值得注意的是，True BASIC允许在一个字符串或注释中使用机器能识别的任意字符。上面已经谈到，ACSII码用七位进制编码可表示128个字符，如果加上bit7，则八位进制可表示256个字符。在IBM-PC机的字符集中就有256个字符（参看附录一）。从字符0到字符127是标准的ASCII字符。这些字符在任何种类的机器上几乎是相同的。但是PC机有一些能出现在屏幕或打印机上的特殊符号，它们对应于字符代码0～31、字符128到字符255是PC机上特有的字符。

2.汉字编码

中文信息处理主要是处理汉字。计算机要处理汉字，就要求将汉字编码。但汉字不象英文，可由基本字母拼出，必须是一字一形，一字一码，因此，要求较大的编码量。目前计算

机中使用的汉字编码是用两字节编码，即用16位二进制数表示。为了与其他编码相区别，国家标准规定的两个字节中的第八位与第十六位均置1。由于汉字编码只在机器内部使用，用户不必了解汉字的具体编码与汉字间的关系。

§1-4 BASIC语言的发展和True BASIC的特点

BASIC语言是程序设计语言中简单、易学、受用户欢迎的语言。20世纪70年代以来，迅速发展的微型机，都配有这种语言，使得BASIC成为世界上应用最广泛的一种计算机语言。但在利用内存空间、编辑功能以及实现程序结构化方面比不上PASCAL和C语言。BASIC语言创始人John G•Kemeny和Thomas E•Kurtz总结了BASIC语言使用经验，吸取了其它高级语言的优点，引入了结构化程序设计的思想，对原有的BASIC语言作了重大修改，于1985年11月公布了新版本的BASIC语言，并命名为"True BASIC"（真正的BASIC）

True BASIC是一种结构化、模块化的语言。它是先进的程序设计语言，它保留了原BASIC简单易学、通俗易懂和适应性强等优点，又在功能上作了重大改进，使它具有结构模块清晰、作图功能强、编译紧凑、运行效率高、程序设计风格优美和可移植性好等特点。同原BASIC相比，具有质的差异。

True BASIC语言的特点归纳起来大致有以下几点：

(1) 具有解释执行和编译执行两种方式，既有利于初学者的交互对话、调试，又可经过编译生成快速执行程序。

(2) 结构控制语句丰富灵活，保证了程序的结构化与模块化，便于程序的分析验证。

(3) 不依赖于具体的硬件，便于移植，可运行于微机、小型机和较大的计算机系统上。

(4) 有功能很强的全屏幕编辑环境，便于程序设计人员灵活迅速地编辑修改程序。

(5) 有较强的图形、动画、音乐功能，适用于开发引人入胜的程序，对初学者及有经验的程序员都有很强的吸引力。

(6) 可直接调用外部子程序或库文件，用户可随心所欲地建立自己的外部子程序库，不经修改便可调用，提供了软件开发的灵活性。

(7) 可充分利用内存，不受64K区段的限制，能以绝对的方式使用IBM-PC的全部内存。

(8) 提供了与DOS操作系统的简练而方便的接口方式，可在程序中直接调用DOS系统的命令和功能，亦可与外部的可执行程序进行连接。

(9) 与已普及的BASIC版本的有较大的兼容性，原有的BASIC程序一般不必做大的修改，便可在True BASIC中运行，在2.03版本中提供了自动变换程序。

(10) True BASIC环境中提供了HELP帮助功能，用户可在需要时使用HELP命令获得系统帮助。

由于以上特点，True BASIC语言对初学者或有经验的程序员都是一个很受欢迎的语言，可以使你得心应手地设计出功能、结构完善的应用程序或软件，而不必另寻其他语言。

§1-5 利用计算机解决实际问题的步骤

1.求解一个比较简单问题

一般要经过以下几个步骤：

(1) 分析问题，明确问题的要求，

(2) 建立数学模型。将一个实际问题（如电路电流计算、导弹轨迹计算、…）用数学语言表示出来（如列出数学式）。

(3) 确定数据结构和算法。解题要考虑数据的组织形式，这就是数据结构。算法是指解题的具体步骤，即考虑好如何一步步地进行操作，一般用流程图来表示算法。

(4) 根据已确定的算法用高级语言编写程序，这一步又称编码 (Coding)。

(5) 调试程序。将写好的源程序上机运行，输入所需的数据（数据应包括不同的“试验数据”）测试在各种不同的情况下程序能否正常运行。如发现程序有错，应修改程序。

(6) 通过调试得到可供正式运行的程序后，上机正式运行，送入正式的数据，得到必要的运算结果。分析结果是否正确。

(7) 整理资料，写出程序文档（程序说明书），包括：题目；任务要求；原始数据；数据结构；算法（可用流程图表示）；程序清单；运行结果；所用计算机系统的配置；操作说明。文档是软件的重要部分，供日后查阅。应养成文档的良好习惯。

2.求解一个复杂的问题

处理一个复杂的庞大的任务，则应按软件工程的要求，进行系统分析、系统设计和程序设计。即应经历以下几个阶段：

(1) 问题定义与需求分析。

(2) 总体设计，或称概要设计。将一个大任务划分为若干个子任务，即划分模块。

(3) 详细设计。根据给定任务（模块）的要求，设计算法和数据结构。

(4) 编写程序和单元测试。

(5) 综合测试与确认运行。

(6) 系统维护。

以上六个阶段为“软件生存周期”其中的 (1) 称为“软件定义期”；(2)～(5) 称为“软件开发期”；(6) 称为“软件维护期”。一般所说的“程序设计”大体包括以上 (3)、(4) 两个阶段。一个大任务不是一个人能完成的，是由系统分析师、系统设计师、高级程序员、程序员共同完成的。

因此，学习True BASIC语言（或其它高级语言），并用它编写程序，只是学习计算机应用的初步，计算机知识浩如瀚海，入门后还要不断扩展自己的知识，以适用面临的复杂任务。

习 题 一

1.一个典型的计算机系统的设备组成包括哪几个部分？

2.什么是源程序？什么是目标程序？计算机如何实现从源程序到目标程序的转换？

3.试说明下列各名词的含义：

机器语言、汇编语言、算法语言、汇编程序、解释程序、编译程序。

4.试述True BASIC语言特点。

5.把下列二进制数转换为十进制数：

(1) 110101　20mm　(2) 11011101　20mm　(3) 101011.111

6.把下列十进制数转换为二进制数：

(1) 65　(2) 289　(3) 65535

7.把下列二进制数转换成八进制数：

(1) 10111101101　(2) 11100111011011　(3) 10011100111001

8.把下列八进制数转换成二进制数：

(1) 541702　(2) 76345　(3) 5667352

9.把下列十六进制数转换成十进制数：

(1) BC5　(2) 3A6B　(3) EFFF

10.把下列八进制数转换成十进制数：

(1) 135　(2) 467　(3) 2777

11.试述计算机解决实际问题的过程。

第二章　磁盘操作系统和怎样使用True BASIC

§2-1 磁盘操作系统的基本功能和启动

一、名辞术语

DOS： Disk Operating System（磁盘操作系统）的缩写。它是管理计算机软、硬件的一组程序，负责计算机各种资源（显示器、磁盘驱动器、打印机、绘图机等）的调度，对各个程序的执行进行管理。True BASIC和其它高级语言都需要DOS的支持。

命令： 告诉计算机去执行一确定活动的词语。例如执行RUN命令是使计算机运行一道True BASIC的程序。

光标： 在屏幕上闪烁的小亮线或块（一个字符大小）。用以提示操作者在光标出现的位置上可键入计算机的符号。

文件： 存于磁盘中的单个程序或数据块。

程序： 以某一种语言书写的指令序列。

提示： 在屏幕上显示一个字符、一个字或一句话，指出应告知计算机现在要作什么。如True BASIC常用的提示符OK，当它出现在屏幕时，则应键入有关True BASIC系统的命令。

窗口： 在屏幕上开设一块显示面积。True BASIC将屏幕分隔成两个窗口：上面的一个可以编辑程序和显示磁盘内的程序；下面的一个可以接收用户键入的True BASIC命令和显示程序运行结果。

磁盘驱动器： 一个外部设备。计算机控制这个设备，可在所装入的磁盘上写入或读取所需要的信息。

磁盘： 磁盘是碟形的磁记录介质，在上面可以存储程序和数据文件。

二、磁盘的基本知识

IBM-PC机一般配有两个软盘驱动器，如PC/AT机配有二个软盘驱动器（标识符为A：和B：）和一个硬盘驱动器（标识符为C：如果有两个硬盘，则另一个标识符为D：）。

通常在微机中使用软磁盘和硬磁盘存放信息，两者存储信息的工作原理和使用方法基本相同。软盘价格便宜、携带方便，深受用户欢迎，因此，我们主要介绍软盘的基本知识。

软盘由外套及盘片组成。外套为永久封闭的硬质保护套，防止磁介质薄膜受损（污染、磁伤等）盘片是上面涂有磁性材料的塑料薄膜圆盘，可在套内旋转。外套上有读写窗口，在读写时，磁头由读写窗对盘片进行读写。

1.磁盘外观与结构

软盘外型见图2-1所示。

(1) 中心大孔为夹持孔，装入驱动器后由轴夹住并带动旋转。

(2) 长条形孔为读写窗，露出磁盘片供磁盘驱动器磁头读写数据用。

(3) 小圆孔为索引孔，每转一圈产生一个索引信号，用以标志磁盘扇区的起始位置。

(4) 防写缺口，用于写控制，对于5.25英寸软盘，如用不透光的物质遮住此口，则禁止对磁盘写操作，故有写保护作用，又称为保护缺口。

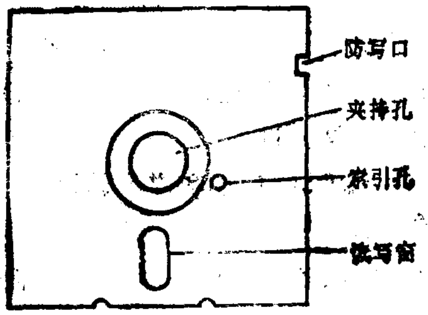

图2-1 软磁盘外形图

2.磁盘的存储格式

在微机中应用的软磁盘通常是5.25英寸(133mm) 软盘，双面容量为360KB，现在还使用1.2MB和1.44MB的软盘。

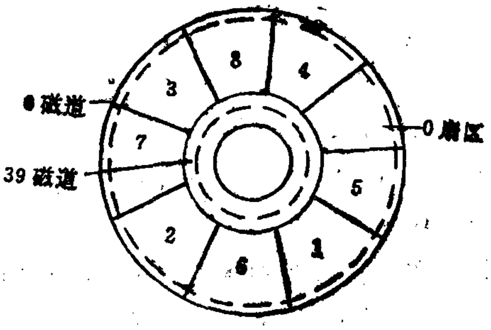

图2-2 磁盘的分区与分道

磁盘使用前，首先要使用操作系统提供的格式命令对其进行格式化。所谓格式化是把磁盘用磁化方法划分成一定的分区格式。DOS 2.0以上版本把磁盘分为40个磁道（0～39）和9个扇区，见图2-2所示。每个扇区的磁道上可存放512个字节信息，因此，磁盘容量可以由其磁盘磁道和扇区算出，对于双面双密度软盘，其容量为：

$$40\times2\times512\times9=368640\text{ 字节}$$

3.磁盘使用注意事项

(1) 不得用手直接触摸读写窗口中的盘面；

(2) 盘片使用后立即放入信封式纸袋，以免落上灰尘；

(3) 不得用重物压在盘片上，或弯曲软盘外套；

(4) 存放时应远离强磁场，以免破坏盘中信息，如远离大电力变压器、电讯、强磁铁等。

(5) 避免日光曝晒，或在温度过高的环境中存放。

二、DOS的基本结构和功能

IBM-PC DOS软盘（磁盘）中核心程序是：

(1) 自检记录程序：存在磁盘的开头处，每次启动时，它最先被读进内存，靠它才能读出DOS盘片中的其它文件。

(2) IBMBIO·COM：设备输入和输出的管理程序，大部分驻留在系统板的8KROM中，称为ROMBIOS，它管理键盘、显示器、打印机、软盘和异步通讯匹配器，并负责与IBM DOS通信。该程序是一个隐蔽文件，列文件时显示不出来。

(3) IBMDOS·COM：一个隐含程序，它的主要任务是管理磁盘存储器和其它系统资源。IBM DOS负责与IBMBIO，COMMAND以及应用程序通讯，启动并控制所有设备的通讯。

(4) COMMAND·COM：一个命令的分析和处理程序，对接收到的信息命令进行分析

然后付诸执行。

DOS的版本很多，从DOS1.0到DOS5.0。后推出的比先推出的作了某些修改和扩充，一般说来后出的兼容先出的。例如在DOS2.0操作系统下编制的程序，在DOS3.3操作系统中也能运行。但是反过来则不一定都能行，这是必须注意的。

四、PC DOS的启动和磁盘转换

1.冷启动

先把PC DOS磁盘插入磁盘驱动器A，关好驱动器门。加电后，PC DOS就自动装入内存，并对内存进行检查，在屏幕上显示出：

```
Current date is fri 1-01-1988
Enter new date: —
```

你应该回答今天的日期。格式是：

mm/dd/yy或mm-dd-yy

分别表示月、日、年。

要求 mm＝1—12；dd＝1—31；yy＝80—99，表示1980—1999。也可以键入四位数，如1980，1999等。

如果你回答日期的格式不对，DOS将显示错误信息。

```
Invalid date
Enter new date(mm/dd/yy):
```

告知输入的日期是非法的，请键入正确的日期。当输入正确的日期后，屏幕显示：

```
Current time is 8:30:14.32
Enter new time: ——
```

你应该回答现在的时间。格式是：

时：分：秒.秒%

要求时＝0－23；分＝0—59；秒＝0—59；秒%＝0—99。

如果输入的时间格式不对，系统就显示：

```
Invalid time:
Enter new time: ——
```

告知输入的时间格式是非法的。在重新输入正确的时间格式以后，DOS显示

```
The IBM Personal Computer DOS
Version 3.20 (C)Copyright International Business Machines Corp 1981,
             1986
             (C)Copyright Microsoft Corp 1981, 1986
A>
```

这里，A>是PC DOS的提示符，表示PC DOS的命令处理程序COMMAND.COM已经准备好，处于接收命令的状态，可以开始打入命令，使用计算机了。

2.热启动

当系统已经加电，正在执行某个程序的时候，如果想启动DOS，可以把DOS盘片插入驱动器A，要同时按下Ctrl键、Alt键和del键，而完成对DOS的重新启动。这种启动又称为系统复位（System Reset），对内存不进行检查。

3.磁盘间的转换

PC DOS的提示符A>，不仅表示系统已经准备好接收命令，还表示当前的磁盘驱动器是A。这时，如果你键入文件名时不指定驱动器名，系统将自动到当前盘A上寻找文件。如果文件在B磁盘上，那么你可以通过命令来改变当前驱动器。

A>　表示A为当前驱动器

A>B:　设定B为当前驱动器

B>　表示B已经为当前驱动器

如果你的IBM PC机同时拥有两个软盘和一个硬盘，DOS将指定软盘驱动器的名字为A和B，而用C来表示硬盘。如果有两台硬盘，另一台用D表示。一个硬盘也可分为C，D两个盘，但它们不是物理盘，而是逻辑盘。

4.文件名

PC DOS中所使用的文件名字由文件名和扩展名两部分组成。文件名由1～8个字符组成，扩展名由1～3个字符组成。文件名的前面还可以有驱动器名。例如，B：Newfile.exe就告诉DOS到驱动器B的磁盘上寻找文件Newfile.exe。它的一般写法是：

[驱动器名：]文件名[·扩展名]

方括号表示其中的内容是可以省略的。下面是一些正确的文件写法：

B:Myprog.com

A:Yourprog.exe

A:Newfile扩展名为空，即没有扩展名。

Newfile缺省当前驱动器名及扩展名。

文件名和扩展名只能用以下一些字符：

A—Z，0—9，$，#，&，@，!，%，( )，0，{ }，-。

另外，有两个特殊的符号：

?　表示可以替换的任意字符；

*　表示可以替换的字符串。

例如：磁盘上有以下文件：

Test1run. exe

Test2run. exe

Test6run. exe

就可以共同表示为Test?run.exe.

Text1run.exe

Test2run. exe

Testall. exe

可以共同表示为Test*.exe。

*.*表示磁盘上的所有文件。

5.文件管理

一个磁盘上可以存放很多文件，如何实现按名查找呢？为此，在磁盘上特定区域建立一个文件目录，文件目录中存放有文件名，文件记录的起始地址、文件长度、文件的建立或修改日期与时间。要查找磁盘上有什么文件，用DIR命令查找此目录就行了。当你建立或删除

文件时，DOS系统就通过目录进行管理。

(1) 目录结构　PC DOS操作系统的2.0以上版本为了改进文件系统的管理，均提供了树型（分层）目录结构，见图2-3所示。

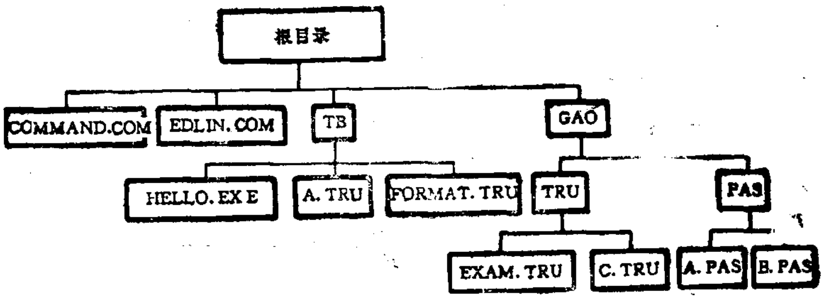
根目录
COMMAND.COM
EDLIN.COM
TB
GAO
HELLO.EXE
A.TRU
FORMAT.TRU
TRU
PAS
EXAM.TRU
C.TRU
A.PAS
B.PAS

图2-3　DOS中的文件层状目录结构

一个软盘文件系统中，根目录下的文件数是有限的，一般双面盘最多112个，超过此限制时，会破坏文件系统。在文件系统中，子目录也是作为文件进行管理的，所以在根目录中建立的子目录与文件的总数也受上述限制。但在子目录中，再建目录与文件时，其文件数仅受磁盘空间限制。使用分层目录可以增加文件系统中的文件数，并且有利于文件的管理与查找。

(2) 路径与路径名

在分层结构的文件系统中，查找文件要指出文件的路径，它指出文件由根目录开始经过的各个子目录。表示这一路径的方法叫路径名（PATH）。路径名的书写方式是：\子目录\子目录\…。

第一个“\”（斜杠）表示是根目录，第二个“\”表示子目录，每个子目录之间均由“\”分开。

例如图2-3中的EXAM．TRU文件的路径名为\GAO\TRU\。

§2-2．PC DOS操作系统的基本命令

一、PC DOS操作系统命令的分类

DOS系统中可将其能接受的命令分为两类，即内部命令和外部命令。

1.内部命令

这类命令之所以称为内部命令，是因为一旦操作系统启动完成后，这些内部命令就和操作系统一起，全部装入内存，只要给出相应的命令名，操作系统便可立即在内存中找到命令，并执行之。

2.外部命令

有些命令不是经常使用的，并且要占用较多内存空间，这些命令就不必要总放在内存中，使用时，只要临时调入内存并执行之，执行完不再在内存中保留。这种命令称为外部命令。外部命令的特征是：①它是在磁盘上可找到的一个可执行文件；②每一外部命令完成一个特定的功能。

由此可见，操作系统可提供各种外部命令，其多少可以不断扩充。另外，只要是可执行文件，均可作为操作系统的外部命令处理，因而各种用户编写的程序，在编译调试后，即可作为外部命令用。凡是具有扩展名·com或·exe或·bat的文件都可以作为外部命令。

二、更换对话盘

提示符为A>时表示与A盘对话（当前盘为A盘），一般说，用户的应用程序盘是放在B驱动器中的，叫作B盘，所以如要改为与B盘对话，就应打入

A>B:

屏幕上相应出现提示符B>，表示B盘为当前盘，可以与B盘对话了。注意，举例用大写字母，并非意味着必须用大写字母不可。打入任何DOS命令时，大小写字母具有同样的功能。能。开机后敲键打出的是小写字母，如愿用大写字母，可用↑（换档键）或Caps lock键来改变大小写字母的显示方法。

当有硬盘D时，键入D：就可以转入与D盘对话。

D盘的提示符是：

D>

当由D盘转为A盘或B盘时也只要相应地打入A：或B：命令就可以了。

当键入错字符时，可按←或Backspace键删去光标前面的一个字符，连按连删，也可按Del键，使光标所在的字符删去。注意，按任何键的时间不要超过0.5秒，如果超过了，就相当于每秒按8次的速度连续按键，即用这类方式按→或←键不放手时就可以快速地移动光标。

注意，在DOS的监控状态下，每当键入回车键时就会把此回车键前面的字符串作为命令或程序文件来执行，如果这个字符串不是DOS命令语句，也不是可运行的文件名，屏幕上就给出错误信息。

三、常用内部命令使用

1.列文件目录 DIR

A>DIR

列文件目录的命令是DIR，在A>状态下键入DIR即可列出A盘中的全部可显示的文件目录。注意，有一部分隐含文件（如自举文件）的文件名不能在DIR命令下显示出来。

在DOS的磁盘中可以调出基本（不是全部）的文件目录见表2-1。

表2-1中每个文件名都有一个后缀，这是扩展名，扩展号由1～3个字母组成，用来表示文件所具有的不同属性。这些后缀有些是必须在建立文件时键入的，有些是在各种语言的建立或编辑时自动加上的。不同的后缀有不同的含义。例如：

.COM——命令
.EXE——执行
.BAT——批处理
.BAS——BASIC语言文件
.FOR——FORTRAN语言文件
.PAS——PASCAL语言文件
.OBJ——目标文件
.ASM——宏汇编文件
.DAT——数据文件

表 2-1

COMMAND	COM	4959	5-07-82	12:00P	用于键盘命令分析
FORMAT	COM	3816	5-07-82	12:00P	用于磁盘格式化
CHKDSK	COM	1720	5-07-82	12:00P	用于磁盘容量检查
SYS	COM	605	5-07-82	12:00P	用于复制系统管理文件
DISKCOPY	COM	2008	5-07-82	12:00P	用于磁盘复制
DISKCOMP	COM	1640	5-07-82	12:00P	用于磁盘比较
COMP	COM	1649	5-07-82	12:00P	用于文件比较
EXE2BIN	EXE	1280	5-07-82	12:00P	用于改为2进制文件
MODE	COM	2509	5-07-82	12:00P	用于设置打印机或彩色显示器的工作方式
EDLIN	COM	2392	5-07-82	12:00P	用于编辑文件
DEBUG	COM	5999	5-07-82	12:00P	用它可调出EXE和COM的汇编程序
LINK	EXE	41856	5-07-82	12:00P	用于连接
BASIC	COM	11392	5-07-82	12:00P	BASIC的解释文件
BASICA	COM	16768	5-07-82	12:00P	改进版本的BASIC解释文件
ART	BAS	1920	5-07-82	12:00P	美术表演程序
COMM	BAS	4352	5-07-82	12:00P	用于通讯文件

.LIB——库文件

.BAK——后备文件

文件目录中每个文件名后还有一串数目字，表示文件占用的字节数以及该文件建立的时间：月-日-年；时；分；上午或下午（A或P）。这是机器自动建立和记录的。

在DIR命令后还可以加以选择项。选择项设置在符号/的后边，其作用如下：

DIR/P　显示文件名到全屏幕满一页时可暂停，以便观察。打入任何键可再继续显示。

DIR/W　用宽面的方式显式，即每行显示五个文件名，但不显示容量大小、建立时间等其它内容。如在DIR后再键入某一文件名（包括后缀）就可以单独把该文件的容量、日期等显示出来。

在列文件、复制文件等命令中，符号＊可代替任何一个字符串，例如键入

```
DIR *.BAS
```

就可以显示出所有以BAS为后缀的文件名。与此类似，符号？可以代替任意一

如果用户的磁盘是插在B驱动器中，而B又不是当前盘，要列出其中的文件时，只要键入

```
DIR B:
```

就可以了。如果想先转入与B盘对话，键入B：再键入DIR，也可列出B盘中的文件名。

2.显示文件 TYPE

如果想要在屏幕上显示出来某一文件的内容，可键入：

```
TYPE 文件名.扩展名
```

屏幕上就会连续显示出这个文件的内容。

注意：有些文件是以字符形式存储的，可以用TYPE命令将其显示出来；有些文件则是以机器码形式存储的，如用TYPE命令就只能显示出一堆无规则的图形符号来。

3.删除文件 DEL 或 ERASE

如果想删除某一个文件，只须键入：

```
DEL 文件名.扩展名
```

或

```
ERASE 文件名.扩展名
```

即可。此后再用DIR命令列文件目录时，这个文件名字就不再出现了。如用代替符号＊删除磁盘的所有文件时，屏幕将出现反问，要求予以确认。例如键入

```
ERASE B: *. *
```

屏幕反问

```
Are you sure(Y/N)?
```

回Y或N表示肯定或否定。

4.改文件名 RENAME

如想将某一文件名为×××＊·×××[文件名·扩展名]的文件名改为○○○○·○○○，可用RENAME命令，只须键入：

```
RENAME ××××·××× ○○○○·○○○
```

5.复制文件 COPY

在已格式化了的新盘上或已有其它文件的旧盘上复制某一文件时，可将载有原文件的磁盘插在驱动器A中，待复制的磁盘放在B驱动器中，然后打入以下命令

```
COPY 文件名 扩展名 B:
```

这个命令执行之后，B盘中就存入了所复制的文件。所用文件名与原来的一样，如果复制时想改用另一文件名，那么只要在上面命令的B：后面加上所需要的新文件名就可以了。

如果想复制一组所有以LW开头的文件名，就可以用以下命令

```
COPY LW*.* B:
```

注意：一个＊代替一个字符串，对于文件名最多可用8个字符，对于扩展名最多可用3个字符，更多的字符虽不会被拒绝，但只有文件名的前8个和后缀的前3个字符可被识别。

6.目录管理命令MD、CD、RD

(1) 建立层次目录结构的命令MD，例如：

MD/User在根目录下建立一个子目录User

MD/User/Joe在子目录User下建一个分子目录Joe

建立层次目录结构只能从根目录一级一级地往下建，先建子目录，再建分子目录，不能两个目录一起建，即一条命令只能建一个目录。

(2) 显示层次目录的命令CD，例如：

CD/User 改变当前目录到子目录User

CD/User/Joe 改变当前目录到分子目录Joe

如果到了Joe分子目录，再键入DIR命令即显示Joe分子目录下的文件和程序名单。

(3) 删除层次目录的命令RD，例如：

RD/User 删除子目录User

删除命令只能删除层次目录中最底层的一个，例如RD/User/Joe命令，只能删掉Joe分子目录，不能删掉User子目录。还要特别注意，在删除层次目录名之前，必须把该目录下的全部内容删除掉，即在该层次目录下键入ERASE *.*，否则删不掉层次目录名。

四、常用DOS外部命令的用法

1.格式化一个新盘 FORMAT

如果想从A盘中复制某一个文件或某几个文件到B盘中去，但若用来复制的B盘是一块新盘，就应先通过FORMAT命令对此新盘做格式化处理，格式化时先把新盘放在B驱动器中，DOS盘放在A驱动器中，在A>提示符下，键入

```
FORMAT B:
```

经过7～8秒钟后，这条命令的执行就完成了，先在屏幕上出现下列提示：

```
?ter new diskette for drive B:
and strike ENRER when ready
```

然后显示：

```
Format copmlete
    1213952 bytes total disk space
        23040 bytes in bad sectors
    1190912 bytes available on disk
Format another(Y/N)?
```

对此做出相应回答即可。

注意：FORMAT命令执行时，对磁盘起总清除作用，如果是对一块旧盘进行格式化，其结果是该盘上的所有文件被清除，变成一块格式化了的空白盘。

如果试图对一块写保护的磁盘进行格式化操作，将出现如下信息：

```
A>FOEMAT B:
Insert new diskette for drive B:
and strike any key when ready
Formatting…
attempted write-protect violation
Format faliure
Insert new diskette for drive B:
and strike any key when ready
```

2.复制磁盘 DISKCOPY

对某些重要的磁盘，如DOS盘的原版盘，最好不要经常使用，而应另外复制一块或几

块，平常就使用这些复制盘，为此就要进行整盘的复制。这个操作十分简单，只需将原盘放在A驱动器中，将待复制的盘（可以是全新的盘，也可以是无用的旧盘）放入B驱动器中，然后键入：

```
DISKCOPY A：B：
```

就可以了。在复制过程中，先对新盘做格式化（Formatting）处理，然后从A盘中读出所有的内容，再记录到B盘中去，这些都是自动进行的，当复制终了时，屏幕上就会出现：

```
COPY another(Y/N)？——
```

对此做出相应的回答就可以了。

如果系统只有一个磁盘驱动器，操作过程稍有不同，但只要按照屏幕提示的步骤去做，就能顺利地完成复制操作。

五、其它命令

1.建立批处理命令 BATCH

如果要把几个命令一起执行，可以利用批处理，批处理文件（Batch）的后缀为·BAT。例如要在开机后立即转入BASICA中的SAMPLES程序，可以建立一个专用批处理文件名为AUTOEXEC·BAT，具体如下：

```
A>COPY con: AUTOEXEC.BAT
BASICA SAMPLES
∧Z
```

有了这个程序在DOS盘中，那么在机器启动后就自动转入执行BASIC A中的SAMPL-ES程序了。

在批处理文件中可以插入PAUSE命令，使运行时到此暂停，此时可做更换磁盘的操作，磁盘换好后，只须键入任何键就可以继续执行。在批处理文件中也可插入REM命令，加上注释语句，使程序运行到此时屏幕上显示相应的说明。

2.显示及改变时间 DATE，TIME

为显示月、日、年或时、分、秒时，只须键入DATE或TIME即可。当屏幕显示相应的数据时，同时还会询问是否要改变，如要改变数据，就按要求格式打入相应的数据，否则直接键入回车键可以退回监控状态。

3.合并文件 copy+

如想把A盘中多个名为M_1，M_2，……（包括扩展名）的文件合并成名为Mn的文件，存到B盘中，则键入

COPY $M_1+M_2+\cdots\cdots$ B:Mn

4.打印屏幕内容

IBM PC机有一个简单的命令，可以一次打印输出当前屏幕上的全部内容，为此只须同时按换档键↑和Prtsc键（新改进的微机，仅需按Prtsc即可打印屏幕内容）就可以了。如果想在打印机上打印出当前屏幕的图形内容，可利用DOS中的Graphics文件。或利用打印机的驱动程序，如2024p.exe，A1lqp.exe等，即可完成屏幕上图形的打印。

5.启动打印机

如果想用打印机打印输出某一文件，可同时键入Ctrl+Prtsc或Ctrl+P，再键入TYPE命令和文件名（包括扩展名）。上面的Ctrl+Prtsc或Crtl+P命令键的作用是在屏幕上逐

行地显示，同时也在打印机上打印输出该文件全部内容。如在显示打印过程中再用这些命令键时就停止显示与打印。这里“＋”号表示同时按下“＋”号两边的键。

执行打印机输出命令时，如果打印机没有准备好。例如，电源没接通或纸已经走完等，屏幕上将出现以下信息：

```
        Device fault
或      Out of paper
```

6.暂停运行　Ctrl＋Namlock或Pause

当运行一个较长的文件时，或是在屏幕上连续显示或打印机连续打印输出一个较长文件的时候，如果希望要暂停一下，只要同时按下Ctrl＋Nnmlock两键或Pause键，就可以实现，此时，只须按任何一个字母，数字或空格键，即可使之继续运行下去。

7.中断运行　Ctrl＋Break或Pause

在程序运行中，或是连续的显示或打印输出一个文件的过程中，想要中断这个过程，使之不再进行下去时，只要同时按下Ctrl＋Break或Ctrl＋Pause两键即可。此后机器退回到监控状态。

8.互换大小写字母 Caps Lock

机器开动以后，键入的字母都是以小写字母形式出现在屏幕上的。如想改为大写形式，有两种方法：一是每次都与键入的字母同时按换档键↑，这和操作打字机相似，但是当字母很多时，这样做就很麻烦；二是，只要按一次Caps Lock键，则以后显示的就都是大写字母了，此键是字母大小写的开关键，所以再按一次就又换成小写字母。

所有的DOS命令，BASIC命令和语句以及其它高级语言，如无特别申明，都是不分大小写，也就是说，不论大写小写机器都同等看待、同样执行。这是因为在程序中，一般都有字母大小写自动转换的功能，但在编释用户程序时应注意大小写字母的差异，因为同样的英文字母，大写和小写分别对应的是不同的ASCII码，可以有不同的作用。

9.方式设置 MODE

这个命令用来设置打印机，或彩色图形显示器，或异步通信接口的工作方式。有以下四种选用方式：

(1) 设置打印机每行打印字符数为80字或者132字，命令格式为：

```
MODE LPT#:[n][,m][,p]
```

其中#表示第几台打印机、可选1，2，3。n是每行打印字符数，可为80或132。m表示每英寸打印机行数，可为8或6，默认值是6，P表示出现超时错误时，规定继续再试的次数。

(2) 对于彩色显示器的工作方式可用以下命令设置

```
MODE[n][,m][,T]
```

n是40或80，是显示器每行显示字符数。第二个选择m可为R或L，使显示字左移或者右移。

如有选择项T，则屏幕将出现反向，问是否已调整合适，回答Y或N，如答N，将允许重复进行左右移字符，直到满意为止。

(3) 对异步通信接口的设置。命令格式为：

```
MODE COM#  :baud[,parity[,databits[,stopbits[,P]]]]
```

其中#是1或是2，表示异步通信口的代码

baud——波特率，可选的数值为110，150，300，600，1200，2400，4800或9600

注意：以上两项是必写的，以下各项可以不写，如不写就按默认值执行。

parity——奇偶校验，可选N，O，E。N是不校验，O是奇校验，E是偶校验，默认值是E。

databits——数据位数，可选7或8，默认值是7。

stopbits——停止位，可取1或2，波特率110时默认值为2，波特率为其它值时默认值为1。

P——这是当异步通信口用于连接打印机时所必须的。其它情况若打了这个字符将出现timeout错误信息。这时只好按Ctrl＋Break使之停止，修改为不带P的命令，然后方可执行

(4) 使一并行打印机的接口接到一个异步串行接口上，用以下命令：

```
MODE LPT#:=COMn
```

#是1，2，3的任一个数，表示第几号打印机。n是1或2，表示通信口的代号，用此命令将使所有传给#号并行接口打印机的信号由异步通信口输出。

注意：①如要用这条命令，必须先按3方式设置好该异步通信口的命令，如果接收设备是打印机，也必须加P参数。

②如果取消这条命令时，应键入命令MODE LPT#:[n][,m]，即重新设置一次打印机的每行字符数及每英寸打印行数。

10.重新启动 Ctrl＋Alt＋Del

当机器陷入某种死循环或因其它原因不能回到监控状态，将会有很长时间不能出现提示符，或者有闪亮的光标，但是对任何键盘命令毫无反应，这时只好重新起动机器方能恢复正常工作。为此可同时按上Ctrl＋Alt＋Del三个键，机器开始重新自举，进入监控状态。这种重新起动的过程是不经过自检的，所以比较快。更严重的系统崩溃（死机）情况对此命令没有反应，遇到这种情况，只有关掉电源，经过10秒后再重新合上开关，重新起动。

上述18条命令是用户经常使用的命令，此外，还有一些DOS命令，它们的使用方法可参阅有关DOS手册。

§2-3 汉字操作系统的功能与应用

一、汉字操作系统

PC DOS操作系统是西文操作系统，只能处理西文字符。为了用于汉字处理（输入、显示、打印等），必须对西文操作系统加以改造，使之成为既能输入处理西文，又能输入处理汉字的中西文兼容的操作系统。CCDOS操作系统就是能实现中西文兼容的操作系统。要实现汉字的输入，必须对汉字进行编码，不论采用何种编码输入方法，应该在处理汉字时使用同一代码（内码）。要实现汉字的输出，必须根据同一汉字代码，能找到代表该汉字的字形数据，在屏幕上显示该汉字的字形。所以汉字操作系统，主要作用是实现汉字的输入与输出的控制与变换。

二、汉字操作系统的组成

因为要实现中西文兼容，所以汉字操作系统中应保留原西文操作系统的三个文件，即：BIOS.COM，DOS.COM，COMMAND.COM。此外还应加入实现汉字输入输出的CCBIOS与汉字字形数据库。CCDOS加入了两个可执行文件与一个字库文件，它们是FILE1.EXE， CCCC.EXE与CCLIB。

其中 FILE1.EXE——用于检查CCLIB库文件是否存在，并为它开辟一个内存区，并初始化接口；

CCCC.EXE——用于装入字库CCLIB，并把CCBIOS装入内存。

所以，汉字操作系统是在装入DOS后再执行一个AUTOEXEC.BAT文件后建立的。AUTOEXEC.BAT的内容是：

```
REM PLEASE WAIT
ECHO OFF
CLS
FILE1
CCCC
ECHO ON
```

三、汉字操作系统的启动

汉字操作系统的启动方法，与西文DOS的启动相同，可以冷启动也可以热启动。在已建立西文DOS时，可在A驱动器中插入CCDOS盘，再用Ctrl+Alt+Del三键启动。

四、汉字操作系统中的拼音输入法

CCDOS 2.1操作系统中提供了四种汉字输入方法：区位码输入法，一个汉字由四位阿拉伯数字编码(用数字0～9)；首尾码输入法，根据汉字象形文字特点，将汉字分解为汉字字根（首尾码），由26个英文字母分别代表不同字根字母组成的编码；拼音码输入法，由汉字拼音组成编码；快速输入法，由首尾码及首音码组成。由于拼音输入法简单易学，这里只介绍拼音码的使用方法。

1.输入状态变换

在启动汉字操作系统后，系统处于西文输入状态，若要输入汉字，需要改变输入状态。使用ALT键和功能键F1，F2，F3，F4，F6之一可改变输入状态；

ALT+F1改为区位码输入状态；

ALT+F2改为首尾码输入状态；

ALT+F3改为拼音码输入状态；

ALT+F4改为快速码输入状态；

以上四种控制均可实现中文输入，此时在屏幕下方会出现输入提示行，在提示行中显示输入状态、输入汉字编码及对应汉字。若要由汉字输入状态改为西文输入状态(ASCII码)，可用ALT+F6改换。

2.汉字拼音输入法

CCDOS 2.1规定输入一个汉字不超过三键。为简化拼音输入，对拼音声韵母作如下规定：

(1) 汉语拼音中的声母用英文中的小写字母输入

b p f d t l g k h j q x z c s r y m n w

(2) 韵母分别用a i u o e v(v代表ü)

(3) 对于合成声韵母，为了简化输入，特作如下规定：

合成拼音韵母	zh	ch	sh	an	ang	ao	ai	en	eng	ing	ong
输入代替字母键	a	i	u	j	h	k	l	f	g	y	s

要熟练掌握拼音法，必须记住上述12个特殊键代表的合成声韵母，否则无法正确地输入汉字。由于汉字中重音字较多，故在输入一个汉字的拼音码后，在提示行中会出现较多同音

字，这时要用数码选择法选出提示行中有编号的汉字，即再输入一个汉字旁的数码，便可选出所需汉字。

例：用拼音法输入“初”字

•先用ALT+F3键改变输入状态为拼音输入状态（并确认为小写英文字母状态）。

•“初”字的拼音为chu，应键入“i”，“u”两键此时提示行内显示：

拼音iu 0：初 1：出 2：樗 ……9：楚 [064]其中0～9代表10个重音汉字数码，[]中为尚有的重音汉字字数。如果要选中“初”字，再输入数字“0”，则“初”被选中并显示于屏幕上的光标所在位置。

如果所要汉字不在当前提示行中，可用 $\boxed{\overset{>}{.}}$ 键向后查找，或用 $\boxed{\overset{<}{,}}$ 键向前查找，找到后，再输入其汉字旁的数码即可。

如果当前显示行的汉字已经选用过，则可直接输入新的汉字，（如这时还要选择提示行中的“出”字)，此时 $\boxed{\overset{>}{.}}$ 与 $\boxed{\overset{<}{,}}$ 键失效，则需使用恢复当前行的操作，用ALT+ $\boxed{\text{数码}}$ 两键可重选当前提示行中的汉字，如果要恢复（提示行）前后页，可用下述方法：

- $\boxed{\text{ALT}}$ + $\boxed{+}$ ——恢复后页
- $\boxed{\text{ALT}}$ + $\boxed{-}$ ——恢复前页

恢复前后页操作后，便又可用数字选择其中所需汉字了。

§2-4 怎样使用 True BASIC

一、True BASIC的使用环境

为了使读者能全面地掌握True BASIC系统的使用方法，在学习有关True BASIC的上机方法之前，我们首先介绍一下True BASIC系统的使用环境。

True BASIC系统需要如下的硬件和软件环境：

(1) 运行于IBM-PC及其兼容机上；

(2) 要求版本在DOS 2.0以上的操作系统支持；

(3) 要求True BASIC Version 1.0/2.03；

(4) 要求系统至少有192 KB的内存容量；

(5) 需要一个双面软盘驱动器；

(6) 配有一台IBM单色或彩色显示器。

二、启动True BASIC

1.英文启动

前提条件是屏幕必须显示出C>或A>提示符，才能启动True BASIC，否则，计算机不接收键盘输入进去的命令。

你应把带有True BASIC执行程序盘放在驱动器A中，并键入Hello。

```
A>Hello
```

稍等片刻，见到屏幕底部出现下列字样：

```
Version 1. 0
Copyright (c) 1985 by True BASIC, Inc.
Published by Addison—Wesley Publishing Company, Inc.
OK
```

“OK”是True BASIC的提示符，在这儿可以键入True BASIC命令。

屏幕上的窗口

在进入True BASIC状态之后，屏幕分为上下两个区域，上面的区域叫编辑窗口（editing window），供用户编辑自己的True BASIC程序用。下面的区域叫背景窗口（history window），供用户输入True BASIC命令，或显示程序运行结果。由于它能给出人机对话期间的痕迹，故取名背景窗口。见图2-4所示。

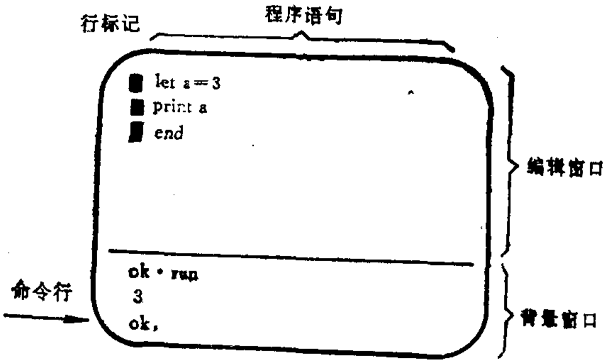

图2-4　背景窗口

(1) 窗口转换。两个窗口，一个可以输入True BASIC命令，一个可以编辑用户程序，光标可以在两个窗口上来回转换。按$\boxed{F_1}$键，光标移至编辑窗口；按$\boxed{F_2}$键，光标移至背景窗口，而且仍回到它离开时的位置。

(2) 窗口大小。编辑窗口系统既定值是17行，背景窗口是7行。中间横线占1行。利用如下命令，可以重新定义编辑窗口大小：

```
OK. split n
```

$(0 \leqslant n \leqslant 24)$

表示编辑窗口为n行。

退出True BASIC

退出True BASIC可键入：

```
OK. bye
```

显示：A>—

表示机器已退出了True BASIC回到DOS状态，此时，可以把True BASIC盘从驱动器中取出，关机结束。

获得帮助

当你键入内容有错时，屏幕上出现What? (Please type HELP.)，这是要你更正错误。

还可以在屏幕上从True BASIC系统中获得相应的解释，这就要键入HELP命令或按一下F10键并回车。回车后，True BASIC将显示相应解释，可得到处理特定问题的帮助。阅读之后，按任一键又会回到编辑窗口，可继续你的编辑。在使用True BASIC时，任何时候按下F10键都会在屏幕上得到有关的帮助。

2.启动带汉字的True BASIC

在A>或C>提示符下，把电子工业部六所（或科海）中文DOS装入A盘，关上驱动器

的门。

```
A>AUTOEXEC
```

如果你用的当前盘不是A盘而是C盘，则在C>提示符下，键入下列命令

C>A：AUTOEXEC（因为中文DOS在A盘，故需在AUTOEXEC批处理命令前加上A：）屏幕上显示下列字样：

```
CCBIOS 3. 00
中国电子工业部第六研究所  1984年8月
C>
```

表示可以启动汉字的True BASIC。如果你仍然用C盘作为当前盘，而且带有True BASIC的执行程序的盘片放在A驱动器，那么键入下列命令

```
C>A：HELLO
```

回车后，在命令窗口中的命令行OK提示符下键入

```
OK. Mode hires
```

命令窗口显示：

```
Ture BASIC here.
Version 1. 0
Copyright (c) 1985 by True BASIC, Inc.
Published by Addison-Wesley Publishing Company, Inc.
OK. mode hires
OK.
```

即可在编辑窗口输入汉字。

三、True BASIC的命令和功能键

1.命令表

表2-2列出True BASIC与BASIC命令对照，并加以注释。

表2-2 True BASIC与BASIC命令对照表

True BASIC命令	BASIC命令	注 释
HELLO	BASIC或BASICA	进入True BASIC和BASIC状态，标志是在屏幕出现OK提示符
BYE	SYSTEM	退出True BASIC和BASIC状态，回到DOS状态，标志是出现C>或A>或B>
OLD文件名	LOAD“文件名”	装入文件，以显示出程序为标志
SAVE文件名	SAVE文件名	存入磁盘一个新文件
REPLACE文件名	SAVE“文件名”	磁盘内有此文件，修改后从新存入，True BASIC用REPLACE命令，而BASIC不管新旧文件一律用SAVE存入或再存入
UNSAVE文件名	KILL“文件名”	删除磁盘一个文件，但KILL要求写全文件名
FILES	FILES	显示文件目录
RUN	RUN	运行当前程序并显示结果

（续）

True BASIC命令	BASIC命令	注　释
NEW	NEW	清除内存中的当前程序
DO NUM		加上行号
DO UNNUM		去掉行号
DO RENUM	RENUM起始新行号、行间距	分配新行号
COMPILE	C>BASCOM C>LINK（即编译和连接后生成带EXE后缀的执行文件）	编译当前文件，True BASIC是在OK提示符下直接进行编辑，编译后用SAVE存入带TRC后缀不能显示的文件，它带有保密性，类似于带EXE后缀文件，不过它是在OK 状态下执行；而BASIC的编译文件是在DOS提示符下执行
LOCATE 字或短语		显示程序所有出现的字或短语，在修改大程序时甚为有用
CHANGE 旧字，改后的新字		把程序中所有不要此旧字地方都换成改后的新字，改成的标志是显示××CHANGE MADE，此命令在修改大程序时甚为有用
TRY旧字，改后的字		同CHANGE的功能，但是它可作到有选择地修改，即用户同意改的字在CHANGE? 后键入Y，否则键入N
INCLUDE文件名	MERGE “文件名”	在当前文件中联接一个磁盘文件，此文件放在光标所指行的后边，联接成功的标志是显示××LINES，INCLUDE在大程序中，常用以合并程序
COPY起行—止行拷行		把程序中从起行到止行一段内容拷到拷行的后面
MOVE起行—止行，指定行		把程序中起行到止行一段内容移到指定行后面
DELETE 起行—止行或函数和子程序名	DELETE起行—止行	删除起行—止行的内容或删除此段函数和子程序
KEEP STARBURST		保留程序中指定的子程序，而其余的均被删除
DO FORMAT		把程序整理成True BASIC的标准形式
MARK		给模块，函数和子程序作标记名
TO 行号或子程序名或模块名		光标移到指定行或子程序或模块的第一行
DO文件名		与RUN类似不同点是在DO后必须指出所运行的文件名
HELP		提供True BASIC中的一些问题的屏幕解释
BREAK行号		在指定行号设置断点，程序运行到此行暂停，供修改和检查内容
CONTINUE		因设断点程序暂停，键入此命令恢复运行
KEY		给键定义新含义
LIST或LIST起行—止行	LLIST或LLIST 起行—止行	打印当前程序清单或打印从起行到止行内容
SPLITn		修改编辑窗口与命令窗口分界线，n为划分参数（0≤n≤24）

此外，还有一些语句，也可当作命令，在命令窗口OK提示符下键入这些语句。这些可作命令的语句有二类，见表2-3，表2-4。

表2-3　任何时候都可作为命令的语句

ASK	BOX	CLEAR	GET	INPUT
LET	LINE INPUT	PAUSE	PLAY	PLOT
PRINT	SET	SOUND		

表2-4　运行一程序后可作为命令的语句

CALL	CLOSE	DRAW	ERASE	MAT
OPEN	READ	RESET	RESTORE	WINDOW
WRITE				

2.功能键

IBM PC机有二种键盘：一种有84个键。其中F1～F10键在键盘的左侧；另一种101个键，F1～F10键在键盘的上部。在True BASIC状态下，这些键的作用见表2-5所示。

表2-5　功能键 F1～F10

键　名	功　　能
F1	移光标到编辑窗口
F2	移光标到命令窗口
F3	光标移到命令窗口，并显示FIND:
F4与F5	块拷贝，在编辑窗口上将光标移到要拷块的上部标志位上，按F4键，然后将光标移到拷块的下部标志位上，按F4键，再将光标移到需要拷贝的上一行的标志位上，按F5键完成整块拷贝
F6	在命令窗口按F6键，可重复以前键入的命令
F7	显示用DEL键删除的行内容
F9	起RUN的作用
F10	起HELP的作用

此外，还有一些用字符标志的光标移动键和编辑键，它们的作用分别列入表2-6和表2-7。

表2-6　光标移动键

键名称	作　　用
Left(或←)	左移1符号
Right(或→)	右移1符号
Up (↑)	上移1行
Down(↓)	下移1行
Home	移至窗口顶部
End	移至窗口底部
PgUp	向上翻页
PgDn	向下翻页
TAB	左移1个字
Shift-TAB	右移1个字
Ctrl-Left	移至行首
Ctrl-Right	移至行尾

表2-7　编辑键

键名称	作　　用
Backspac (空格键)	删除光标左边1字符
Del	删除当前字符
Ctrl-Home	删除光标左边1字
Ctrl-PgUp	删除当前1字
ESC	删除光标左边行部分
Ctrl-End	删除光标右边行部分
Del	光标在行标志处按Del键删除一整行内容
Ctrl-F	改变前景颜色
Ctrl-B	改变背景颜色
Ctrl-E	改变边缘颜色

习　题　二

1.为什么必须有操作系统？为什么要用操作系统启动计算机？冷启动与热启动的区别是什么？

2.用你使用的计算机，正确输入你当前的时间（月、日、年和时、分、秒）。

3.用DIR命令列出A盘和C盘文件目录，并按文件名后缀区分文件的属性。

4.用COPY命令把C盘某一文件拷贝到A盘上，并用DIR命令验证此文件拷贝的结果。

5.用DEL命令删除第4题中拷贝到A盘上的文件，用DIR文件名命令验证此文件删除的结果。

6.用TYPE命令显示磁盘某一文件的具体内容，并启动打印机，把该文件内容打印在打字纸上。

7.用MODE命令设置打印机的打印规格（字符数行宽）。

8.用MD命令建立自己的子目录，并将True BASIC的执行程序HELLO.EXE拷入该子目录。

9.启动英文和启动带汉字的True BASIC执行程序有何区别？编辑窗与背景窗的作用是什么？

10.在哪种状态下，使用DIR和FILES命令，两者有何区别？

11.用SPLIT命令可改变True BASIC的编辑窗口和命令窗口界线，在OK提示符下，键入SPLIT 5时，中英文两种状态下窗口的变化是否相同？

12.用什么命令可由DOS状态进到True BASIC状态，或由True BASIC状态退回到DOS状态。

13.SAVE和REPLACE命令有何区别？利用这二个命令往磁盘试存某一个文件。

14.用COMPILE命令编译一个True BASIC程序，考查一下编译前和编译后的程序运行上有何不同。

15.如何用拼音输入一个汉字？如何查找同音字？如何恢复当前提示行中的汉字？

第三章　True BASIC 的基本概念及基本语句

§3-1　基本概念

一、True　BASIC程序的构成

用True BASIC语言编写的程序称为源程序，简称True BASIC程序。

例1 已知a，b，c三边之长，计算任意三角形面积。计算公式：

$$S=\sqrt{p(p-a)(p-b)(p-c)}$$

式中，$p=(a+b+c)/2$，$a=8$，$b=4$，$c=6$

源程序及运算结果：

```
! the True BASIC program
LET a=8
LET b=4
LET c=6
LET p=(a+b+c)/2
LET S=sgr(p*(p-a)*(p-b)*(p-c))
PRINT "s=";s
END
OK. RUN
S=11.619
```

从上面的例子可以看出，一个True BASIC程序是由语句行和注释行两大部分组成的。注释用以帮助人们阅读程序和提高程序的清晰度，在程序中是可有可无的。

通过上面的例子，可以看出构成True BASIC程序有如下规则：

1.一个程序是由若干个语句组成

例1程序是由7个语句组成。语句的多少要根据具体问题的复杂程度而定。每个程序必须以END结束。

2.一个语句由定义符和语句体组成

(1) 定义符　如上面例子程序中有定义符LET，PRINT，END等，它们是语句的核心。它规定了计算机执行什么任务。

(2) 语句体　即计算机执行的具体内容。上面例子中a=8就是通过LET命令计算机将8存入a单元。又如p=(a+b+c)/2是将(a+b+c)/2的值存入p单元中。还有其它的语句将在以后各章节中一一介绍。

二、True BASIC字符集

True BASIC 字符是True BASIC 语言的符号集合，该字符集中的字符是书写 True BASIC语言程序的最基本元素，见表3-1所示。

表3-1 有确定意义的字符

字符	含意	字符	含意
	空格（输出为一个空白）	!	感叹号或注释符
=	等号或赋值号	&	字符串连接运算符
+	加号	,	逗号
-	减号	.	句号或小数点
*	乘号或星号	;	分号
/	除号或斜线	'	单引号
^	乘幂符号	:	冒号
(	左括号	?	问号
)	右括号	<	小于号
%	百分号	>	大于号
#	井号	"	双号或字符串分隔符
$	美元符号或字符串类型说明符	_	下划线

True BASIC字符集包括：

(1) 字母字符 26个大小写英文字母（A～Z，a～z）。

(2) 数字字符。10个数字字符（0～9）。

(3) 一些有确定意义的字符，见表3-1所示。

(4) 对于已经汉化的True BASIC版本，还可以使用汉字。

应当注意的是在书写True BASIC程序时，决不允许出现True BASIC字符集以外的任何字符，诸如希腊字母α、β、γ和π等。

三、程序中的常数表示

常数是程序运行中始终保持不变的值，它可以分三大类：数字常数、逻辑常数和字符串常数。其中逻辑常数是0或1（0代表“假”或“不成立”，1代表“真”或“成立”），以下我们着重讨论True BASIC数值常数和字符串常数。

1.数字常数

有整数、小数及指数。其中指数表示形式与数学表示形式不同。如数学写1×10^3,$5\times10,^3$ 7.8×10^{-5}；True BASIC则写成1e3，5e3，7.8e−5。

True BASIC语言对数的表示范围没有限制，但具体到一个计算机系统上，就不能不有

所限制了。一般，IBM-PC机中True BASIC值取范围是1.11254e－308到3.59539e＋308。凡小于下限的数都按零处理；大于上限的数都按上限值处理（即按3.59539e＋308处理），并给出溢出信息（OVERFLOW）。True BASIC有效位数（十进制）为十位。如果数小于有效位数时，则用日常记数的方法；如果大于有效位数时，则用指数形式表示。

2.字符串常数

字符串常数（简称字符串）是True BASIC语言中一种很有用途的非数字型常数，它是由一对双撇号括起来的字符序列。双撇号只起到界限符的作用，相当于数学中的括号。字符串常数最长可达32766个字符（汉字16383个）。如"How are you"，"中国人民万岁"，"A="，"Lilen"等。

使用字符串常数需要注意以下几点：

(1) 空格也可作为字符或字符串使用。如" "如"a bo^{o}k"，在a与b之间有一空格。

(2) 引号本身" "也可以作为一个字符使用。如"x""y"表示字符串由三个字符组成（即x，" "，y组成）。

(3) 在字符串中对于大小写字母是有区别的。如"ABC"和"abc"表示两个不同的字符串。

四、程序中的变量表示

在程序运行中可以改变的量，称为变量。True BASIC程序中使用的变量与数学中的变量含义有所不同，数学中的变量隐含着“未知”；而程序中的变量是一个被赋了值的变量名。也就是在程序运行中给它分配一个以该变量名为标记的存储数据的单元，该单元的值可随时变化。所以定义一个变量名实际上相当于给一个存储单元具体地址。因此在一个程序中不能用同一个变量名表示不同的变量。

True BASIC规定变量名可以用字母、数字和底线符号“—”组成。但必须由英文字母开头。比如X1，t，sum，X2y，time-of-day等都是合法变量名。而#-of，3xt，x+y，time of day，α，β等都是不合法变量名。

变量可分数字型变量量字符串变量。

1.数字型变量

又分简单变量和下标变量。

(1) 简单变量　变量之间无联系的变量，如前面的引用合法变量名皆为简单变量名。

(2) 下标变量　带有下标值的变量，用以表示数组中的元素。下标变量在数学中写成x_1，x_2，x_3，在程序中则写成x(1)，x(2)，x(3)。下标值是写在圆括号中。

2.字符串型变量

分为简单字符型变量和下标字符变量。其变量名也要符合前面的规定，但在变量名后面必须增加一个“$”字符，如x$，name$(1)，name$(2)，name$(3)等。

注意，凡是True BASIC的专用名词不能作为变量名使用。

五、程序中常用的内部标准函数

True BASIC中的标准函数是由语言本身提供的内部程序，故又称内部函数。这些标准函数可以大大简化程序的设计，例如：

SQR(x)被称为平方根的函数，x是一个表达式，也称为函数参数。SQR(9)是将9这个参数值代入求平方根函数，其返回值（函数值）为3。

表3-2 常用内部标准函数

函　数	功　能	备　注
SIN(X)	$\sin x$	自变量以弧度为单位
COS(X)	$\cos x$	自变量以弧度为单位
TAN(X)	$\tan x$	自变量以弧度为单位 [X接近$(2n+1)\pi/2$时，TAN(X) 可能会超过机器所允许的上限值而溢出]
ATN(X)	$\mathrm{arctg}\,x$	$-\pi/2<$函数值$<\pi/2$
PI	π	$\pi=3.141592$
LOG(X)	$\ln x$（自然对数）	$x>0$
LOG10(X)	$\lg x$（常用对数）	$x>0$
EXP(X)	e^x	$e=2.71828$
SQR(X)	$\sqrt{x}$	$x>=0$
ABS(X)	$\lvert X \rvert$	
INT(X)	求小于或等于X 的最大整数	如　INT(−5.6)=−6 INT(3.8)=3 INT(0.4)=0
SGN(X)	1　X>0 sgn(X)=　0　X=0 −1　X<0	符号函数
RND	产生(0,1)之间一个均匀分布的随机数	

表3-2中仅列出了True BASIC的一些常用内部标准函数，有关其它一些True BASIC内部函数，我们将在以后章节中介绍。

六、程序中的表达式

表达式是用运算符和括号将常数、变量和函数连接一起的式子。表达式有两种类型，即即数值表达式和字符串表达式。

1.数值表达式

又分算术表达式、关系表达式及逻辑表达式。

(1) 算术表达式：它与数学运算符相对应，只是有的写法不同。

数学运算符+、−、×、÷、乘方对应True BASIC的运算符为+、−、*、/、∧。

如$2ab^3$写成2 * a * b∧3。它们的运算顺序有优先级的规定：

（ ），函数，∧，*，/，+，－，其优先级依次下降。

如数学表达式

$$\frac{2\cdot \mathrm{SIN}^5(3x)}{3}+8$$

写成True BASIC表达式，下面所标数码表示优先顺序。

```
2*SIN(3*x)∧5/3+8
④ ②   ①   ③⑤⑥
```

再举几个例子：

数学表达式	算术表达式
$2\pi R$	2*pi*r（或2*3.141592*r）
$v_0t+\frac{1}{2}at^2$	v0*t+a*t∧2/2
$\sqrt{p\cdot(p-a)\cdot(p-b)\cdot(p-c)}$	sqr(p*(p-a)*(p-b)*(p-c))
$\frac{x+y}{x-y}$	(x+y)/(x-y)

(2) 关系表达式：用于改变程序的流向。它是由关系运算符连接两个算术表达式。运算结果为逻辑值，即“真”或“假”，其运算符为：

运算符	含义	表达式形式
=	等于	x=y
<>或><	不等于	x<>y或x><y
>	大于	x>y
<	小于	x<y
>=或=>	大于或等于	x>=y或x=>y
<=或=<	小于或等于	x<=y或x<=y

(3) 逻辑表达式：用于改变程序的流向。它是由逻辑运算符连接两个关系表达式。运算结果也是逻辑值，即“真”或“假”。其运等符为：

运算符	含　义	表达式形式
NOT	逻辑非	NOT(1=1 AND 2=2)
AND	逻辑与	x>6 AND y>7
OR	逻辑或	x=7 OR y=1

下面举几个例子：

①D>=C AND x>y　当D≥C和x>y时，此逻辑表达式值为真。

②x=0 OR y=0　当x=0或y=0时，逻辑表达式值为真。

③L1>=L2 OR L　当L1≥L2或L为真时，逻辑表达式值为真。

④A>=B AND NOTW

当A≥B，且NOT W为真时，表达式值为真。其它情况均为假。

⑤A+B>=C+D AND E=F　当A+B≥C+D以及E=F时，表达式值为真。

逻辑运算也有优先级顺序：

NOT→AND→OR。举例说明：

如a=3，b=4，c=5逻辑表达式如下：

$$\underbrace{\underbrace{\underbrace{\underbrace{c-b}_{\text{①}}=a}_{\text{②假}}\ \text{AND}\ \underbrace{\underbrace{b-a}_{\text{①}}=1}_{\text{②真}}}_{\text{④假}}\ \text{OR}\ \underbrace{\text{NOT}\ \underbrace{\underbrace{8*b-a}_{\text{①}}>\underbrace{c/2}_{\text{①}}}_{\text{②真}}}_{\text{③假}}}_{\text{⑤假}}$$

2.字符串表达式

它由字符串常量或字符串变量用运算符&连接而成的，字符串常量和字符串变量是字符串表达式的特殊形式。

设a$=″MO″，b$=″men″，c$=″tum″

(1) 连接字符串常量：

″MO″&″men″&″tum″.

(2) 连接串变量：

A$&B$&C$

(3) 连接字符串常量与串变量：

“MO” &B$& “tum”

(4) 连接字符串与子字符串：

子字符串即字符串中的一部分字符。

①设F$=″good morning! ″

″very″&″ ″&F$[1:4]

其结果为：very good

②″AMA″&„ZED″[2:3]

其结果：AMAED

③(″AMA″&″ZED″)[2:6]

其结果：MAZED

④″AMA″&″ZED″

结果为：AMAZED

字符串关系表达式及逻辑表达式中的运算符与算术表达式运算符一样，只是比较的方法采用了字典方式。这种字符顺序是以ASCII码为序的。见附录1。

下面举几个例子。

表达式	结果
"a"<"b"	真
"a"&n$<"b"	假
"hi"[1:1]>"fox"	真
"a"<"a"	真
1=1 or 3+2=6	真
1=1 AND 3+2=6	假
a$<b$ or b $>=a$	真
a$="lee" OR NOT a $="lee"	真
NOT(1=1 AND 2=2)	假

§3-2 基本语句

True BASIC基本语句有输出语句（PRINT），赋值语句(LET)，读语句（READ），置数语句（DATA），键盘输入语句（INPUT），恢复数据区语句（RESTORE）。

一、输出语句（PRINT语句）

输出语句PRINT，用来在屏幕上显示字符串和数字。它是True BASIC语言中使用最频繁、用法最灵活的一种语句。

1.PRINT语句一般格式为：

PRINT[〈变量〉/〈常数〉/〈表达式〉][，……][，/；]

其中，[ ]为任选项；〈变量〉/常数/〈表达式〉表示一项中可为三种情况之一；[，……]表示若有多项显示项时，各项间用逗号分开，继续写另外的数据项；[，/；]为语句末尾可选用标点符号。

2.PRINT语句的功能：

PRINT语句具有运算与输出双重功能，它对表达式先进行运算，然后将结果输出到终端。

PRINT语句输出的基本型有两种：数字型及字符型。

(1) 输出数字型的常数，已知变量的值及表达式的值。程序及运行结果为：

```
LET b=6
PRINT 8
PRINT b
PRINT 3*b+2
END
OK. run
8
6
20
```

(2) 输出字符串常数（原样照印字符串）。已知串变量的值及表达式的值。程序及运行结果为：

```
!也可以输出字符串常数、变量及表达式的值
LET a$='this is a string.'
```

```
PRINT"this is a book."
PRINT a $
PRINT"I say: "& a$
END
OK. RUN
this is a book.
this is a string.
I say: this is string.
```

3.PRINT语句的基本输出格式

(1) 输出各项用逗号“，”分隔的称为固定格式。它是按固定的区段输出。区段的划分与计算机类型有关。IBM及各兼容机的规定每行分为5个区段。如果输出的项数超过5段时，则机器会自动将其余各项转到下一行，仍按固定格式输出。

(2) 输出各项用分号“；”分隔的称为紧凑格式。即输出的各项紧靠在一起。但对数字来说各项数据之间留有符号位置。同样，输出的项数超过一行所限定的字数（一行为80个字符）时，机器会自动换行并仍以紧凑格式输出。

数字型程序及运行结果为：

```
PRINT 2, -5, 4, 7, 9, 20, -8  !用逗号间隔数据项，则按固定格式输出
PRINT 2; -5; 4; 7; 9; 20; -8  ! 用分号间隔数据项，则按紧凑格式输出
END
OK. run
2        -5        4        7        9
20       -8
2  -5  4  7  9  20  -8
```

字符串型程序及运行结果为：

```
! 字符串型输出格式
PRINT"a", "b", "c", "d", "e", "f", "g"
PRINT"a"; "b"; "c"; "d"; "e"; "f"; "g"
END
OK. RUN
a        b        c        d        e
f        g
abcdefg
```

使用紧凑格式输出一组词组时须注意词与词间设置空格。请看下列程序及运行结果为：

```
PRINT "this is a book."
PRINT"this"; "is"; "a"; "book."
PRINT"this", "is", "a", "book."
END
OK. run
this is a book.
this is a book.
this      is      a      book.
```

(3) PRINT语句末尾标点符号用法

①当PRINT语句末尾带分号“；”时，表示本语句中的各项输出完后不换行。下一个PRINT语句的第一个输出项，接着在本行按紧凑合式输出。请看下列程序及运行结果为：

```
PRINT 1; −2; 3;
PRINT 4; −5; −6;
PRINT“Good-bye”
END
OK. RUN
1 −2  3  4 −5 −6 Good-bye
```

②当PRINT语句末尾带逗号时，表示本语号中的各项输出完后不换行。下一个PRINT语句的第一个输出项，接着在本行按固定格式输出。请看下列程序及运行结果为：

```
PRINT 1, −2, 3,
PRINT 4, −5, −6,
PRINT“Good-bye”
END
OK. RUN
1         −2          3          4          −5
−6        Good-bye
```

③当PRINT语句末尾不带标点符号时，表示本语句各项输出完后，自动换行。请看下列程序及运行结果为：

```
PRINT 1, −2, 3
PRINT 4, −5, −6
PRINT “Good-bye”
END
OK. RUN
1          −2          3
4          −5          −6
Good-bye
```

④当PRINT语句中没有任何输出项时，则输出一个空行。请看下列程序及运行结果为：

```
PRINT 1; 2; 3
PRINT
PRINT 4; 5; −6
PRINT “Good-bye”
END
OK. RUN
1 −2  3

4 −5  6
Good-bye
```

4.利用TAB函数定位输出

PRINT语句的输出方式相当灵活，除了上面我们讲过的利用逗号、分号控制输出位置的基本方法外，我们还可以使用输出格式函数TAB（x），直接指定输出项的具体位置。

TAB函数在PRINT语句中的一般格式为：

PRINT TAB (x) ；输出项

TAB函数的功能是将光标移到输出行中由数值表达式x所确定的位置上（数值表达式的值应在1到80之间，即不超过屏幕的边界；若其值为小数，则四舍五入取整），把跟在它后面的输出项显示输出在这个位置上。请看下列程序及运行结果为：

```
LET a=20
LET b=10
LET c=5
PRINT TAB(5); "a"; TAB(a); "b"; TAB(a+15); "c"
PRINT TAB(5); a; TAB(a); b; TAB(a+15); c
PRINT TAB(5); "a+b="; a+b;
PRINT TAB(a); "b+c="; b+c;
PRINT TAB(a+15); "a+c=; a+
END
OK. RUN

  a            b              c
    20           10             5
  a+b=30       b+c=15         a+c=25
```

注意：①使用TAB (x) 格式必须与PRINT语句配合。②括号中的x可以是常数、已知有值的变量或表达式。③x值必须是正整数，若运算结果有小数，计算机会自动取整。④x值只能从左往右增加⑤EXPR是数值表达式或字符串表达式。

二、提供数据语句

给计算机提供数据就是将数据存放到某些变量或数组中，实际上，就是对变量或数组赋值。提供数据大体有三种方式：用LET语句赋值；用READ/DATA语句赋值和键盘输入。

1.赋值语句 (LET语句)

(1) 赋值语句一般格式为：

LET〈变量〉=〈表达式〉

或 LET〈变量〉,〈变量〉,〈变量〉,…=〈表达式〉

其中，LET是语句定义符，True BASIC要求语句定义符必须写，不能省略。“=”为赋值号，意义不同于数学中的等号“=”。变量是数值变量，也可以是字符串变量。

(2) 赋值语句功能：是首先计算出赋值号右边表达式（由常数、已知变量、函数及其组合）的值，然后把这个值[illegible]到赋值号左边的所有变量。

例如：LET R=15

就是“将15送给变量R”或称“使变量R的值等于15”。

再如：LET I, J=100

就是将100分别送给变量I, J，次序是从左到右。

(3) 使用LET语句时，注意以下几点：

①在True BASIC语句中等号为赋值号，它具有动态的含意，表示一种逻辑操作，即将赋值号右边表达式运算结果通过LET命令存放在以变量名为标志的单元中。因此赋值号的左边必须是变量，右边也可以是常数，已知变量或表达式。赋值号两边不能调换。比较下面两个程序可以看出赋值号的作用。程序及运行结果为：

```
LET a=1        LET a=1
LET b=2        LET b=2
LET b=a        LET a=b
PRINT a; b     PRINT a; b
END            END
OK. RUN        OK. RUN
1    1         2    2
```

由于第3行a与b对调，致使两程序输出结果不同。

②表达式中的变量必须是已赋过值的，否则计算机将解释为零。

③一个LET语句可以给一个或多个变量赋值（多个变量要求相同赋值时才可用）。

2.键盘输入语句（INPUT语句）

键盘输入语句是使变量取得数据的另一种方法。赋值语句（LET语句），不管是赋予常数、变量或表达式，变量所取得的值是确定的。在实用中，经常要求在程序执行中由用户从键盘将常数赋给变量，这可增加程序的灵活性和通用性。INPUT语句具备以上特点。

(1) 键盘输入语句的一般格式为：

INPUT [Prompt〈提示字符串〉：] 〈变量表〉

其中，可选项[Prompt〈提示字符串〉：]被选用时，在语句执行时，首先在屏幕上显示提示字符串，作为提示信息；〈变量表〉为输入的变量名，可以是一个变量，也可以是多个变量。当有多个变量时应以逗号将变量隔开。如果省略选择项[Prompt〈提示字符串〉：]，则执行该语句时，将在屏幕上出现“？”，然后等待用户从键盘上输入数字或字符给相应的变量。

(2) INPUT语句的功能：是当程序运行到INPUT语句时，在屏幕上显示一个问号，用户在问号后键入数据。当用户输入完数据后，计算机立即将这些数据依次赋给INPUT语句中对应的变量，然后继续执行下面程序。

例如：

```
INPUT a, b, c
LET x=a+b+c
PRINT a; b; c
PRINT "x="; x
END
OK. RUN
? -3,  5,  1
-3  5  1
x=3
```

如果选择了〈提示字符串〉项，那么我们就在字符串提示下键入数。

例如：

```
INPUT PROMPT "RADIUS=?" : R
LET S=PI * R^ 2
PRINT "S="; S
END
OK.RUN
RADIUS=? 40
S=5026.55
```

(3) 使用INPUT语句时应注意以下二点：

①INPUT语句无运算功能。因此键入的数据只能是常数(整数或小数)，不能是表达式。

②在INPUT语句中要求给多个变量赋值时，要求键盘在一行内输入数量相等，类型逐个对应一致的常数，常数间应以逗号隔开。若输入数据个数少于变量的个数就回车，则系统显示提示：

```
Too few input items, please add more
```

此时用户应继续在问号后输入数据，如果一行中输入的数据项太多时，则显示：

```
Too many input items, Excess ignored
```

系统自动舍弃多余的输入。

3.读数语句/置数语句 (READ/DATA)

LET 语句只能给一个或几个变量赋值。当遇到大批数据 (如矩阵运算、线性方程组求解以及数据处理等问题)时，就不很方便也不经济，而READ/DATA语句可以成批的给变量输入数据。

(1) 读数据语句/置数据语句一般格式为：

READ〈变量表〉

DATA〈数据项表〉

READ语句中可以有一个变量名或多个用逗号隔开的变量名。

DATA语句为置数据语句。它是非执行语句，不产生任何操作。常数可以自由设置，在多个常量之间要以逗号分隔，而且每一数据行的第一个字必须是DATA。

执行READ语句时，自动地将DATA语句中的各个常量一一对应地赋给READ语句的各个变量。

例如：已知三个系数$a=1$，$b=-3$，$c=2$，求一元二次方程$ax^2+bx+c=0$的根。请看下列程序及运行结果为：

```
READ a, b, c
DATA 1, -3, 2
LET x1=(-b+sgr(b^2-4*a*c))/(2*a)
LET x2=(-b-sqr(b^2-4*a*c))/(2*a)
PRINT "x1="; x1, "x2="; x2
END
OK. RUN
x1= 2           x2= 1
```

(2) 使用READ/DATA语句应注意以下几点：

①READ语句/DATA语句必须配合使用。但不一定成对出现。下面两个程序的功能完全相同：

```
READ x, y, z                READ x
DATA    1                    DATA 1, 2, 3, 4, 5
READ a b,                    READ y, z
DATA 2, 3                    READ a, b
DATA 4, 5                   PRINT x; y; z; a; b
PRIND x; y; z; a; b         END
END                         OK. RUN
OK. RUN                     1 2 3 4 5
1 2 3 4 5
```

②READ中的变量必须与DATA中的数据一一对应、类型匹配，如果数据个数少于变量个数则会出错，显示：

Reading past end of data字样。

③DATA可放在END语句前任何地方，但不能放在END语句之后。

④因为READ语句无计算功能，所以DATA语句中必须是常数，而不能是表达式，如：

DATA 3＊5，sin(2)，7/2是错误的。

4.恢复数据区语句（RESTORE语句）

用READ语句从DATA中读出一批数据后，数据指针已移到这批数据之后，若需将同一批数据再赋给另一批变量是否需要将DATA语句重新写一遍呢？完全不必要了。RESTORE语句可以使数据指针重新指向第一个DATA语句的第一项数据，另一READ语句可再从程序中第一个DATA语句的第一项数据开始重新读取数据。这称为恢复数据区，请看下列程序及运行结果为：

```
READ a, b, c                    READ a, b, c
DATA 1, 2, 3                    RESTORE
READ e, f, g                    READ e, f, g
DATA 1, 2, 3                    RESTORE
READ i, j, k                    READ i, j, k
DATA 1, 2, 3                    DATA 1, 2, 3
PRINT a; b; c; e; f; g; i; j; k PRINT a; b; c; e; f; g; i; j; k
END                             END
OK. RUN                            OK. RUN
1 2 3 1 2 3 1 2 3                  1 2 3 1 2 3 1 2 3
```

在有些时候，仅需要使用DATA语句中的一部分数据，就可以在使用RESTORE语句的同时配合使用“假读”手段，即在程序中设置若干没有实际用途的变量，将要跳过的数据读入这些对程序不起作用的变量中去。请看下列程序及运行结果为：

```
READ A, X, B, X, C, X, D, X, E
PRINT A: B; C; D; E
RESTORE
READ X, A, X, B, X, C, X, D, X, E
PRINT A; B; C; D; E
DATA 1, 2, 3, 4, 5, 6, 7, 8, 9,0.
END
OK. RUN
1 3 5 7 9
2 4 6 8 0
```

实现要求分别打印DATA语句中的奇数项数据和偶数项数据。

恢复数据区语句一般格式为：

RESTORE

§3-3 串变量在提供数据语句中的使用

一、在LET语句中使用

LET语句格式为：

LET〈串变量〉=〈字符串〉

此格式与使用简单变量一样。但这里一个LET语句同样可给多个变量赋值。格式为：

LET 〈串变量1〉，〈串变量2〉，…，〈串变量n〉=〈字符串〉

程序及运行结果：

```
LET a$, c$='IBM PC'
LET b$='IS A MICROCOMPUTER'
LET d$=a$&b$&'.'
LET e$=c$&b$&'.'
PRINT d$
PRINT e$
END
ok.run
IBM PC IS A MICROCO COMPUTER.
IBM PC IS A MICROCO COMPUTER.
```

第一句字符串“IBM PC”同时赋给a$及c$。同样，赋值号的右边也可以是已知串变量或串表达式。

有时为了使用方便True BASIC语言还提供了子字符串。即将原字符串中某部分字符串赋给串变量。此部分字符串简称子串。

子串赋值格式：

LET〈串变量〉[i：j]=〈子串〉

i为原串变量中子串开始位置；j为原串变量中子串结尾位置；等号右边的子串为新子串。

程序及运行结果为：

```
LET n$='abcdefghijklmnopqrstuvwxyz'
PRINT n$
LET n$[1:9]='qweio' ! 将 abcdefghi改成qweio
PRINT n$
END
OK. RUN
abcdefghijklmnopqrstuvwxyz
qweiojklmnopqrstuvwxyz
```

使用子串赋值，还可以把一个子串附加到另一个字符串的开始、中间或末尾，也可以删除字符串中的部分。例如下面程序：

```
LET n$="Elizabeth weaver"
PRINT n$
LET n$[1:9]="Betty"                    ! 替换子串"Elizabeth"
PRINT n$
LET n$
LET n$[17.:max]=".M.D"                ! 将".M.D"附加到末尾
PRINT n$
LET [n$6:5]="k."                      ! 将".M.D"附加到6处
PRINT n$
LET n$[16:50]=""                      ! 删除".M.D"
PRINT n$
LET n$[1:0]="Dr."                     ! 将"Dr."附加到开始处
PRINT n$
END
OK.RUN
Elizabeth weaver
Betty weaver
Bety weaver.M.D
Betty k. weaver.M.D
Betty k. weaver
Dr. Betty k.weaver
```

二、在INPUT语句中使用

INPUT语句格式为：

INPUT〈串变量1〉，〈串变量2〉，…〈串变量n〉

INPUT PROMPT〈字符串？〉：〈串变量〉

PRINT〈字符串〉；

INPUT〈串变量〉

例如

```
INPUT a$, b$
PRINT a$; b$
END
OK. RUN
? Good, morning
Goodmorning
OK. RUN
? Good, " morning! "
Good morning!
```

INPUT语句末尾可以使用逗号，它表示两个变量的数据可以在一行输入：

INPUT 〈串变量〉，

INPUT 〈串变量〉

下面举一简单例子。

```
 INPUT a$,
 INPUT b$
PRLNT a$; b$
END
OK. RUN
? Good, "  morning ! "
Good morning!
```

给串变量输入数据应注意：

①用LET给串变量赋值时应当用双引号括起来。而用INPUT 语句或 READ/DATA 语句时，如果字符串不包含有逗号、冒号和开头空格，可以不用引号括起来，否则必须用双引号括起来。

②在READ/DATA语句或INPUT语句中简单变量和串变量可以混合使用。如：

```
READ a, b, c, d$
PRINT a, b, c, d$
READ a$, b$, c$, d$
DATA 1, 2, 3, 4, 10, 20, 30, "ok! "
PRINT a$, b$, c$, d$
END
OK. RUN
 1        2        3        4
10       20       30       ok!
```

三、在READ/DATA语句中使用

```
READ t$; n$; m$
PRDNT t$, n$, m$
DATA"总数是:" , "二百", "米"
END
OK.RUN
总数是: 二百米
```

四、有RESTORE语句时使用

```
READ a, b, c, d$
RESTORE
READ e$, f$, g$, h$
PRINT a, b, c, d$
PRINT e$, f$, g$, h$
DATA 10, 20, 30, "ok! "
END
OK. RUN
10    20    30    ok!
10    20    30    ok!
```

§ 3-4　几个简单语句

一、注释语句 REM 或！

REM（或！）是REMARK的缩写，它的作用是给程序或语句加以说明（注：从本章开始各程序已经使用了注释语句）。

在程序中它可以与别的语句一样，自占一行，也可在别的语句后边用注释语句加以说明。

注释语句是非执行语句。它与PRINT的功能不同。前者只是在列程序清单时注释内容被原样照写；后者是在程序执行时照原样显示。

二、STOP语句和END语句

STOP语句在程序中起到终止程序执行的作用，即程序执行到该语句时，终止运行。STOP语句可以写在程序中任何需要终止运行的位置。

例如：语句IF a$=b$ THEN STOP，表示当字符串变量A$与B$相等时，终止程序运行。

END语句与STOP语句不同，每个程序的末尾都应以END语句结束，因此END语句的出现标志整个程序的结束。

三、PAUSE语句

PAUSE语句可设置暂停时间，使程序暂停若干秒后再继续执行，通常用于屏幕显示一帧内容，要求暂停一定时间以便观察时。语句格式为：

PAUSE　t

t为一数字表达式，取值为正值，以秒为单位。例如：PAUSE 5（暂停5秒）

四、SOUND语句

SOUND语句可按指定发音时间产生一个指定频率的声音，语句的格式为：

SOUND Freq,Longth

其中，Freq为发声频率，单位为赫（每秒多少周）

Longth为发声长度，单位为秒

例SOUND　1000,1.5　（发出1000赫的声音1.5秒钟）

例2 模拟电话铃声的程序（请读者在学完本书第五章循环结构之后再阅读该程序）：

```
!
PRINT"Press  any key to stop sound! "
DO
   FOR i=1 to 1
       FOR j=1 TO 30
            SOUND 600, 0.03
            SOUND 1500, 0.03
       NEXT j
       PAUSE 2
   NEXT i
LOOP UNTIL KEY INPUT
END
```

五、GET KEY语句

GET KEY〈变量〉语句，可以将从键盘上输入的任意键的值（代码）赋给指定的变量。执行此语句时，等待用户由键盘输入任意键，用户输入一键后，程序将其代码值赋给指定的变量，此值就是所按键的ASCII码值，如值超出ASCII码范围，就是键盘上特殊键代码，通过下面的程序可以了解此语句的用法与功能。

```
PRINT “按任意键[”;
GET KEY Value
PRINT Chr$(Value);
PRINT “]键的值为：“; Value
END
```

GET KEY语句也可以用于控制程序暂停，因为执行GET KEY语句时，一直等待用户输入一键为止，若不关心输入的实际键值，此语句仅起暂停执行后续语句作用。

六、LINE INPUT语句

当输入字符串时，LINE INPUT语句可以允许一次输入一行字符串，其中包括输入空格，逗号，直至遇到回车为止。

LINE INPUT语句的一般格式为：

LINE INPUT [Prompt expr$：]：rarl$，Vars$，…

LINE INPUT语句中可以有一个prompt expr$：，但它只在第一行输入时产生提示，其他输入行只显示“？”

LINE INPUT语句在输入时，每输入一行可存入一个字符串变量，输入中的行首的空格不被舍去，不必要使用双引号，若有也被认为是字符串的一部分。

例3 给出的程序每次要求输入一行字符串，以回车为结束。当不输入字符串而直接回车时，则程序不再接受输入，然后程序打印出输入的各行内容。（请读者学完本书有关章节之后，再阅读该程序）

```
!
SET MODE "hires"
DIM A$(10)
LET n=0
PRINT "每次输入一行!"
DO
   LET n=n+1
   LINE INPUT A$(N)
   IF A$(N)="" THEN EXIT DO
LOOP
FOR I=1 TO n
   PRINT AS(I)
NEXT I
END
```

§3-5 应用举例

本节编写几个True BASIC的简单程序，以巩固和加深理解这些基本内容。

例4 按下列公式编写计算自由落体位移的程序。

$$S=V_0t+gt^2/2$$

程序及运行结果为：

```
INPUT PROMPT "Vo, T=? " : VO, T
LET G=9.8
LET S=VO*T+G*T^2/2
PRINT "S="; S
END
OK. RUN
VO, T=? 3, 2 ↙
S=25.6
```

例5 向一个RC串联电路充电，电容上的电压为：

$$U=U_0(1-e^{-\frac{t}{r\cdot c}})$$

U_0为直流电源的电压。求t=1sec，r=100kΩ，c=1μf时U等于U_0的百分之多少？程序及运行结果为：

```
INPUT PROMPT "t=? ": t
INPUT PROMPT "r=? ": r
INPUT PROMPT "c=? ": c
LET u=100*(1-exp(-t/(r*c)))
PRINT "u="; u
END
OK. RUN
t=? 1
r=? 100
c=? 1
u=.995017
```

例6 设我国1980年工业产值为100，每年平均增长率为6%，计算2000年时的工业产值。计算公式为：

$$x=100\times(1+y)^n$$

式中，y为年增长率；n为年数；x为产值。程序及运行结果为：

```
READ n, y
LET x=100*(1+y)^n
PRINT "x="; x
DATA 20, 0.06
END
OK. RUN
x=320.714
```

习 题 三

1. 把下列各数由日常记数法转换成指数记数法，或由指数记数法转换成日常记数法。

(1)375833　　(2)0.000000001

(3)0.99999999　　(4)735811294

(5)2.171E+10　　(6)3.4E−3

(7)0.1E−5　　(8)256E+3

2. 下列符号哪些是True BASIC的合法简单变量名？并指出非法变量名的错误所在.

(1)A$　　(2)J*K　　(3)π　　(4)X_Y_Z

(5)H$C　　(6)C$ $　　(7)A_B$　　(8)3$

(9)a　　(10)AB　　(11)X(1)　　(12)3R

(13)X12　　(14)B♯　　(15)PI　　(16)IF

3. 将下列数学表达式写成True BASIC的表达式.

(1) $x+a(b-dy)$　　(2) $x+\sin|x|+\sin|y|$

(2) $(x^2+y^2)^2\cdot\sqrt{e^{b-1}}$　　(4) $3^x\cdot\sin^x+\mathrm{tg}x$

(5) $\dfrac{(2\cdot\sin45°)^2}{e^2\cdot\lg5}$　　(6) $\dfrac{\sin45°+\cos45°}{\sin45°-\cos45°}$

(7) $\dfrac{x+y}{x+\dfrac{x+y}{x-y}}$　　(8) $\dfrac{2\cdot\mathrm{tg}x}{1+\mathrm{tg}x}$

(9) $\dfrac{\pi}{2}(1+e^x+e^x)$　　(10) $\omega\cdot(\sin\alpha+\sin\beta)$

4. 指出下列True BASIC表达式的运算顺序.

(1) 2+5*SIN(5/3)^2/2+3

(2) 2*ABS(3^2/3-2)/5

(3) COS(2*SIN(2+SQR(2^3)/4)/5/2)/4

(4) EXP(6)/SIN(X)/4.8*8^2

5. 指出下列数学表达式所对应的True BASIC表达式中的错误，并加以改正.

数学表达式	True BASIC表达式
(1) $\dfrac{a-1}{b+1}$	A−1/B+1
(2) $(xy)^5$	XY.5
(3) $\sqrt{p(p-a)(p-b)(p-c)}$	$\sqrt{p*(p-A)*(p-B)*(p-c)}$
(4) $\omega(\sin\alpha+\cos\beta)$	ω*(SIN(α)+cos(β))
(5) $\dfrac{-b+\sqrt{b^2-4ac}}{2a}$	(−B+SQR(B²−4AC))/2*A

6. 下列语句写法中有哪些错误？

```
(1)x∧2=4
   x+2=∨−3
(2)READ A; B; 3a; D3F; x$; FF$
   DATA 3; 8; 10; "ABC", 6, "CAD", 3/8
(3)INPUT x; y; z
   IPPUT PROMPT"A$="; A$
   INPUT A=3*5
(4)PRINT    A=3*5
```

7. 写出下列程序的运行结果

(1)
```
READ a, b, c, d, e, f
LET t=a+b+c+d+e+f
LET u=t/f
PRINT # 1: "u="; u
DATA 8, 4, 2, 1, 3, 5
END
```

(2)
```
READ a, b
DATA 1, 2, 3
READ b, c, d, e
RESTORE
DATA 4, 5, 6, 7
READ d, e, f, g, h, i, j, k, l, m, n
RESTORE
DATA 8, 9, 10
PRINT: a, b, c, d, e, f, g, h, i, j, k, l, m, n
END
```

8. 试编写用PRINT语句输出下列结果的程序:

(1)x+y=2*3=6

(2)一个九九乘法表。

(3)一个"中"字.

9. 编写程序求如下两种情况的三角形面积:

(1) 当输入底边a，高h时，面积 $S=\frac{1}{2}ah$。

(2) 当输入两个角t_1，t_2和夹边a时

$$S=\frac{1}{2}a^2\frac{\sin t_1\cdot\sin t_2}{\sin(t_1-t_2)}\quad(t_1, t_2\text{为弧度})$$

10. 编一程序求下述梯形面积:

(1) 上底6cm，下底10cm，高6cm的梯形面积。

(2) 输入五组上底a，下底b，高h的值求各种梯形面积。

第四章　分支结构

§4-1　概　述

True BASIC程序运行的基本规则是按顺序执行。但是，在许多实际问题中，人们经常需要计算机按不同的情况进行不同的处理，也就是程序运行的顺序要根据具体情况改变。例如：

(1) 输入－15，－13，7.2，－9.8，100，83，77，1，208，－16，十个数，要求计算机选出其中的负数及其和，并将它们打印出来。

(2) 输入x值，要求输出y值，y是x的函数。

$$y=\begin{cases} 7.25 & (\text{当}0\leqslant x<10.5) \\ 12.66 & (\text{当}10.5\leqslant x<19.5) \\ 42.55 & (\text{当}19.5\leqslant x<50) \\ 62,41 & (\text{当}55.5\leqslant x<65) \end{cases}$$

对于上述问题，都需要计算机对给定的条件进行分析、比较和判断，并按判断后的不同情况转移到程序中不同的语句去进行不同的处理，这种程序结构称之为“分支结构”。

True BASIC为实现分支结构，提供了IF THEN分支控制结构和SELECTCASE控制结构。在介绍上述两个分支控制结构之前，首先介绍一种常用的编程工具——程序流程图。

§4-2　程序流程图

在程序设计中，为了说明解决问题的算法和逻辑结构，常常要使用程序流程图。

流程图具有形象易懂等优点，它不仅可以指导编写程序，而且还可在调试程序中用来检查程序的正确性。流程图还能作为程序说明书的一部分提供给别人，以便帮助别人理解你编写程序的思路和结构。

一、常用的流程图符号

在流程图中常用的图形符号，见图4-1所示。

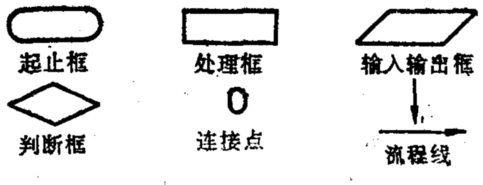

图 4-1

其中，起止框：用于表示程序的开始或结束；处理框：表示一个操作或几个操作的处理；输入输出框：表示显示或打印输出以及数据的输入；判断框：用于表示程序的分支，有一个入

口，两个或多个出口，框内标有判断的条件，根据条件是否成立选择一个出口；连接点：表示流程图的出入口，利用连接点可以实现流程图之间的连接；流程线：表示程序的走向。

二、流程图应用举例

下面结合一个例子说明流程图的使用。

例1 假设一个质点作匀加速直线运动。初速度为v_0，加速度为a，则t时刻质点运行的距离是$S=v_0t+\frac{1}{2}at^2$。现假定$v_0=10\text{cm/s}$，$a=5\text{cm/s}^2$，要求每隔1秒钟计算一次S，并打印该时刻的t和S值，直到S≥500cm时停止。

1.分析题意，确定计算方案

首先应给 v_0、a 和t输入具体的数值，然后计算t时刻的S值，并打印输出相应的t和S的值，接着应进行一次判断，看看S≥500是否满足。如果条件满足，就终止计算，转结束；否则，t的值增1，再重新计算S值，打印输出对应的t和S值，判断条件S≥500是否满足，…，直至条件满足为止。

2.编制流程图

根据上面分析可画出流程图4-2。

图4-2流程图中每一个框（除开始框外）表示一段程序（包括一个或多个语句）的功能。各框内清楚地写明了要做事情，简单明确。

从这个例子可以看到流程图的作用，它把要做的各项工作和执行顺序，以及在什么条件下应该做什么，表示得明白无误，即使不作解释，也能对其作用和流程一目了然，根据这个流程图去编写程序就不困难了。

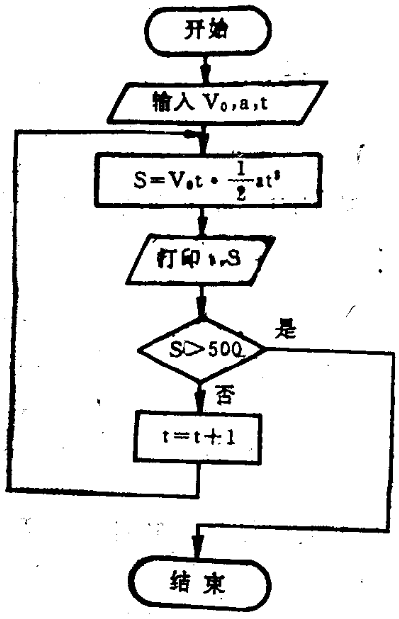

图 4-2

§4-3 IF-THEN 分支控制结构

分支控制结构使程序的执行具有选择性，即可根据条件来选择不同的处理，这种控制结构对程序设计十分重要，可使程序具有逻辑判断能力，因而使程序具有灵活性和智能性。

True BASIC的IF语句有三种格式。

一、单行分支结构格式

IF＜条件＞THEN＜语句1＞[ELSE＜语句2＞]

其中，条件可以是任何逻辑表达式，取值只有逻辑值（真或假）。

这种形式的IF语句的进行过程是：若条件为真，则执行THEN后面的语句；否则执行ELSE后面的语句。有时这种IF语句形式不包括ELSE部分，这时如果条件为假，将直接执行此IF语句后的下一个语句。

例2 计算运费。设货物运输价格P（元）与运输距离s（公里）有关。按以下公式定价（P为吨/公里运价）。

$$P=\left\{\begin{array}{ll} 20 & (\quad S<100) \\ 17.5 & (100\leqslant S<200) \\ 15 & (200\geqslant S<300) \\ 12.5 & (300\leqslant S<500) \\ 10 & (\quad S\geqslant 500) \end{array}\right\}$$

输入S和货物吨数W，要求计算出总运费。

COST（总运费）＝P＊W＊S，请看下列程序及运行结果为：

```
INPUT Prompt"enter distance and weight"; S,W
IF  S<100 THEN P=20
IF  S>=100 AND  S<200 THEN P=17.5
IF  S>=200 AND  S<300 THEN P=15
IF  S>=300 AND S<500 THEN P=12.5
IF  S>=500 THEN P=10
COST=P*S*W
PRINT "COST=" ; COST
END
OK,RUN
enter distance and weight? 20,100↙
COST=40000
```

二、复杂分支结构格式

单行分支结构中，THEN或ELSE后只能包含一个语句。实际编程时，要求按条件分别执行一些语句（语句块），该格式就是包含语句块的复杂分支结构，其格式为：

```
    IF 〈条件〉THEN

       …
       …  } 语句块1
       …
     [ELSE
       …
       …   } 语句块2
       …
       ]
END IF
```

其执行过程是：当条件为真，则执行THEN下面的语句块1；当条件为假，则执行ELSE下面的语句块2。如果省略了ELSE部分，则当条件不成立时退出IF结构，去执行它下面的语句。应注意的是，复杂分支结构的块结构IF语句后面都必须有END IF语句，ELSE与END IF必须各自独占一行。

三、多层分支控制结构格式

在实际问题中，经常存在由较复杂的条件来决定不同的处理，这时用单一条件的分支控制结构已不能满足要求。对于由多个条件来决定不同处理的情况，我们可以采用多个单条件分支控制来设计程序，但是在程序设计时要注意，每一个IF结构必须有相应的END IF语句相对应，也就是说这种多层IF嵌套时必须保证 IF 与END IF完全配对。使用IF控制结构

嵌套的缺点是，如果分层较多时，程序变得繁琐不清，且易造成END IF丢失。下面我们举例说明这种情况。

例3 用程序实现符号函数：

$$Y=\begin{cases}1 & \text{当}x>0\\ 0 & \text{当}x=0\\ -1 & \text{当}x<0\end{cases}$$

要求程序执行时x值由键盘输入。

对于符号函数的程序实现，就要用两个分支语句来设计。我们先画出实现符号函数的程序流程图，见图4-3所示。

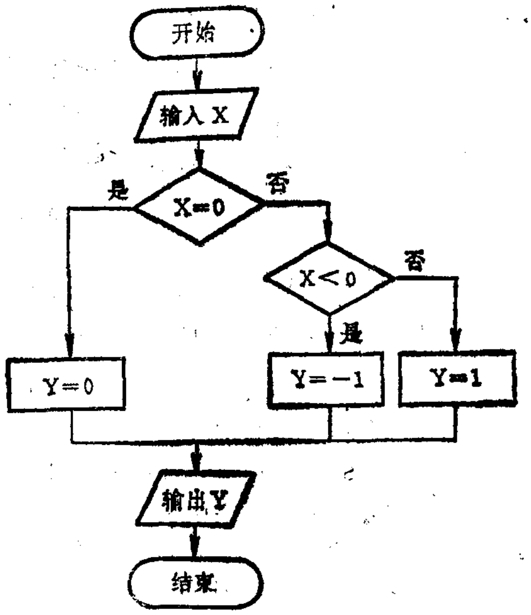

图 4-3

根据图4-3流程图设计的程序和运行结果为：

```
! Function of sign
!
PRINT "Please input x:" ;
INPUT x
IF x=0 THTN
    LET y=0
ELSE
    IF x<0 THEN
         LFT y=-1
    ELSE
       LET y=1
    END IF
END IF
PRINT "y=" ;y
END
OK.RUN
Please input x:-7
y=-1
OK.
OK.RUN
Please input x:3
y=1
OK.
OK.RUN
Please input x:0
y=0
OK.
```

在程序中使用了两个IF…ELSE…END IF结构，实现了两层IF嵌套。实际中我们遇到

的问题常比这复杂得多。为了避免分层较多时，程序结构变得太复杂，易造成 END IF 丢失，True BASIC语言提供了第三种标准形式的IF语句——多层分支选择结构ELSE IF 结构。ELSE IF结构的格式为：

```
IF      〈条件1〉    THEN
                    …⎫
                    …⎬语句块1
                    …⎭
ELSE  IF    〈条件2〉    THEN
                    …⎫
                    …⎬ 语句块2
                    …⎭
ELSE
                    …⎫
                    …⎬ 语句块3
                    …⎭
END IF
```

ELSE IF语句模块可以有任意多个，ELSE部分也可以省略。这是最复杂的 IF 语句形式，它可以使用户的程序按照一定的次序进行一系列的检验。ELSE IF 结构的IF语句执行过程是：对语句中的若干个条件顺序进行检验，一旦其中某一条件为真，则执行其相应的THEN下面的语句块，随后便退出IF结构，不再去测试其它条件了。

现在用ELSE IF结构将上例程序改写为：

例4

```
! Function of sign
!
PRINT "Please input x:" ;
INPUT x
IF x=0 THEN
       LET y=0
ELSE IF x<0 THEN
       LET y=-1
ELSE
       LET y=1
END IF
PRINT "y=" ;y
END
```

从以上三种IF语句的标准格式可以看出，前二个格式都是单条件；最后一个格式是多条件的，形成了多层二分支结构。另外我们还可以发现，第一个格式是一种单行形式，所有的内容都写在一行中；后二个格式是多行结构，整个结构以 IF 语句开始，以 END IF 语句结

束，中间部分的ELSE语句或ELSE IF语句将两个或多个语句分隔开来，书写成的程序段结构非常清晰。

§4-4 情况选择分支控制结构

在程序设计中，常常遇到要在几个可供选择的条件中选择其中之一执行。这种情况，可以使用多重IF-ELSE-END IF语句来实现，但程序结构繁琐，而且层次多时，容易产生错误。True BASIC 语言提供的 SELECT CASE 控制结构，以清晰的结构形式实现这种控制。

一、情况选择分支控制结构格式

```
SELECT CASE 〈测试表达式〉
    CASE 〈测试11〉,〈测试12〉, …
            …┐
            …├ 〈语句块1〉
            …┘
    CASE 〈测试21〉,〈测试22〉, …
            …┐
            …├ 〈语句块2〉
            …┘
    ⋮
    CASE ELSE
            …┐
            …├ 〈语句块n〉
            …┘
END SELECT
```

二、SELECT CASE语句执行过程

1.SELECT CASE语句的执行过程

首先算出SELECT CASE语句中测试表达式的值，然后由第一个CASE后的测试内容进行测试（有多个测试项时，以“,”号分离，并按排列顺序进行），一旦测试条件满足，便执行该CASE下面的语句块，若一个CASE的全部测试都不满足，便执行下一个CASE的测试。若全部CASE都测试不满足，便执行其后的语句。所以，SELECT CASE 是一个单入口，单出口的分支控制结构。

2.测试表达式

测试表达式可以是字符串表达式或数字表达式。如果表达式是数值型的，则后面所有的测试也必须是数值型的；若是字符串表达式，则后面所有的测试也必须是字符串型值。

3.测试内容

CASE语句中的测试项有三种可能的形式：

(1) 字符串或数字型常数，如“A”、“B”，“C”或0，1，2等；

(2) 字符串或数字型常数 TO字符串或数字型常数，如“A” TO “M”；−30 TO 30等；

(3) IS〈关系运算符〉字符串或数字型常数，如IS<200、IS<>“BOOK”等。

在一个CASE语句中允许上述几种测试的混合，例：6，6 to 16，IS>=30。

三、使用SELECT CASE控制结构应注意如下几点

(1) SELECT CASE语句中的测试表达式，可以是数字型或字符串型，并且其取值类型必须与各CASE语句中测试取值类型相一致。

(2) 各CASE语句的测试项应是不相同的。

(3) 在SELECT CASE语句第一个CASE语句之间不允许有任何语句。

(4) CASE ELSE语句不能省略，即使在CASE ELSE与END SELECT 语句之间无任何语句时也一样。

例5 输入一个整数，在终端上显示输出对应输入整数的用英文表示的月份。

程序和运行结果为：

```
! Display months in English.
INPUT prompt "n=" :n
SELECT CASE n
CASE 1
          PRINT "January"
CASE 2
          PRINT "February"
CASE 3
          PRINT "March"
CASE 4
          PRINT "April"
CASE 5
          PRINT "May"
CASE 6
          PRINT "June"
CASE 7
          PRINT "July"
CASE 8
          PRINT "August"
CASE 9
          PRINT "September"
CASE 10
          PRINT "October"
CASE 11
          PRINT "November"
CASE 12
          PRINT "December"
CASE ELSE
          PRINT "Error in input data"
```

```
END SELECT
END
OK. RUN
n=11
November
OK. RUN
n=1
January
OK. RUN
n=14
Error in input data
OK.
```

程序运行时，假如输入一个小于或等于0的整数，或者输入一个大于12的整数，则终端显示输出“error in input data”的输入数据出错的标志。这个程序中CASE语句的测试内容采用了第一种形式，即数字型常量方式进行测试。

§4-5 应用举例

例6 求一元二次方程的根。

(1) 分析题目，进行算法设计。大家知道，一元二次方程$ax^2+bx+c=0$的根是

$$\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$，也就是方程的根由它的三个系数a，b，c决定。

应注意，一元二次方程可能有二个、一个甚至没有实根，这要由判别式$\Delta=b^2-4ac$决定。

如果$\Delta>0$，方程有两个不等的实根。

$\Delta=0$，方程有两个相等的实根。

$\Delta<0$，方程没有实根。

(2) 画出程序的流程图。根据上面的分析，可画出一元二次方程求解的程序流程图，见图4-4所示。

(3) 根据图4-4流程图可编写True BASIC程序及运行结果为：

```
! Solve quadratic equation given coefficients.
!
PRINT "Give coefficients a,b,c:" ;
INPUT a,b,c
LET discr=b^2-4*a*c
IF a=0 THEN
   PRINT "One root at:"
   PRINT "x=" ;-(c/b)
ELSE IF discr>0 THEN
   LET s=sqr(discr)
   PRINT "The roots are:"
   PRINT (-b+s)/(2*a)
```

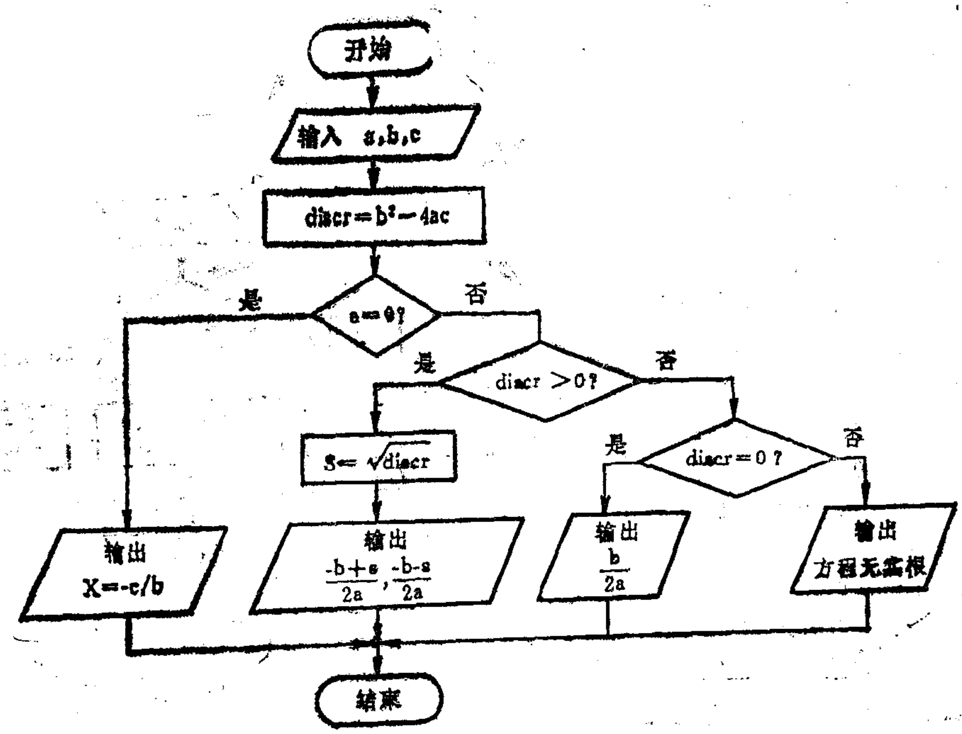
开始
输入 a,b,c
discr=b²-4ac
是
a=0?
否
是
discr >0?
否
S=√discr
是
discr=0 ?
否
输出
X=-c/b
输出
-b+s/2a, -b-s/2a
输出
b/2a
输出
方程无实根
结束

图 4-4

```
PRINT (-b-s)/(2*a)
ELSE IF discr=0 UEEN
    PRINT "The root is:"
    PRINT-b/(2*a)
ELSE
    PRINT "No real roots."
END IF
END

OK.RUN
Give coefficients a,b,c:?1,-5,6
The roots are
    3
    2
```

例7 托运某货物，若货重不超过10公斤，按每公斤0.15元收费；若货重超过10公斤而不超过20公斤，则超出部分按每公斤0.30元收费；若货重超过20公斤，则超出部分按每公斤0.45元收费。如果货物重量为x公斤，试编写True BASIC程序求出运费y。

解 根据题意，可列出费用y与货重x之间的函数关系：

$$y=\begin{cases}0.15x & (\text{当}0<x\leqslant 10)\\ 0.15\times 10+0.30(x-10) & (\text{当}10<x\leqslant 20),\\ 0.15\times 10+0.45(x-10) & (\text{当}x>20)\end{cases}$$

上式是我们在高等数学的学习中常遇到的一类函数；对于自变量x在某个范围内的值，通过一个分析表达式可以得到函数y的值；而对x在另一范围内的值，则通过另一分析表达式得到相应的y的值。这类函数被称为分段函数（piecewise function）。

分段函数求解的程序流程图，见图4-5所示。

根据图4-5流程图编写的程序及程序运行结果为：

```
! Solve piecewise function
!
PRINT “please input x:” ;
INPUT x
SELECT CASE x
CASE 0 to 10
        LET y=0.15*x
CASE 10 to 20
        LET y=0.15*10+0.30*(x-10)
CASE is>20
        LET y=0.15*10+0.45*(x-10)
CASE else
END SELECT
PRINT “x= ”;x; “kg”
PRINT “y=” ;y; “Yuan”
END
OK.RUN

please inout x: 16.6
x=16.6kg
y=3.48 Yuan
```

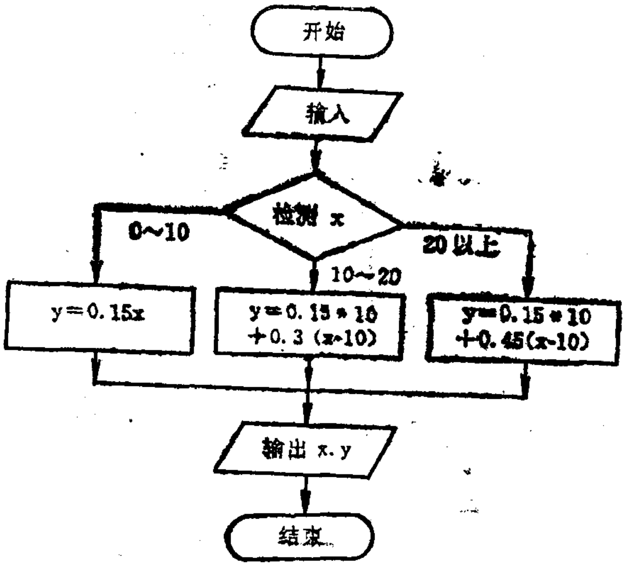

图 4-5

例8 给出公元年数，判断该年是否闰年？

解 判断闰年的方法是：

(1) 公元年数如用4除不尽就不是闰年；

(2) 公元年数如能被4整除，而不能被100整除则是闰年；

(3) 公元年数能被4，100，400整除就是闰年。

根据上面的分析可画出判断闰年方法的流程图，见图4-6所示。

根据图4-6流程图，编写的程序清单及运行结果为：

```
! Decide which year is leap year
!
PRINT “Please input NS:” ;
INPUT ns
IF mod(ns,4)< >0 THEM
    PRINT ns; “is not leap year”
ELSE IF mod(ns,100)< >0 THEN
   PRINT ns; “is leap year”
ELSE IF mod(ns,400)=0 THEN
   PRINT ns; “is leap year”
ELSE
   PRINT ns; “is not leap year”
END IF
END
```

```
OK.RUN
Please IMPUT NS:? 1990
    1990 is not leap year
OK.RUN
Please IMPUT NS:?2000
    2000 is leap year
```

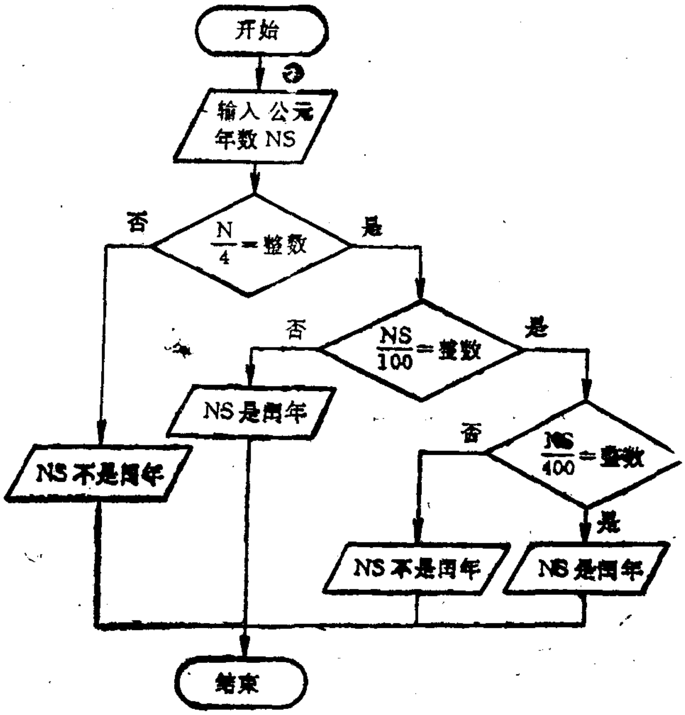
开始
输入公元年数 NS
$\frac{N}{4}$=整数
否
是
$\frac{NS}{100}$=整数
否
是
NS是闰年
NS不是闰年
$\frac{NS}{400}$=整数
否
是
NS不是闰年
NS是闰年
结束

图 4-6

程序中采用了MOD(x,y) 函数用于判别x能否被y整除，大家知道，函数MOD(x,y) 返回x被y除的余数，若x能被y整除，则x被y除的余数为0，所以我们可以通过MOD (x, y) 返回值是否为0来判别x能否被y整除。

习 题 四

1. 什么叫流程图？六种基本的流程图符号的含义是什么？

2. 画出下列问题的流程图：

(1) 输入一个数到计算机，输出这个数的绝对值。

(2) 输入三个数，打印出其中的最大值。

(3) 求1至100之间全部偶数之和。

(4) 打印出1至500之间哪些能被3和5整除的数。

3. 编写计算下列分段函数的True BASIC程序：

$$y=\begin{cases}\cos(x^2-1) & \text{当}x>10\text{时}\\ e^{x+1} & \text{当}x=10\text{时}\\ \sin(x^2+1) & \text{当}0<x<10\text{时}\end{cases}$$

其中x分别取8个点（无大小顺序），即25,0,7,4,13,10,1,22.

4. 给出−5 −13,5.3,−7.1,83,66,−9,1,−308,−10十个数据，编写一个程序把数据中的负数打印出来，然后将负数的和也打印出来。

5. 对于给定的x和y值，计算下列函数z的值。

$$z=\begin{cases}\sqrt{\dfrac{x^2+y^2}{x-2y}} & 当x\geqslant 0, y>0时\\ x^2-|x^2+y|+1 & 当\geqslant 0, y\leqslant 0时\\ \dfrac{\sin(x+y)}{x^2+y^2} & 当x<0, y>0时\\ \dfrac{\cos(x-y)}{x+y} & 当x<0, y\geqslant 0时\end{cases}$$

6. 任给十个数$a_1, a_2 \cdots\cdots a_{10}$，试计算:

(1) 平方和:

(2) 乘积;

(3) 其中正数的个数及总和:

(4)其中负数的个数及总和。

7. 对下列各题编程序，计算并输出n值，直到$S \geqslant 10^{-8}$。

(1) $S=1^2+2^2+\cdots\cdots+n^2$

(2) $S=1+(1+2)+(1+2+3)+(1+2+3+4)+\cdots\cdots(+1+2+\cdots+n)$

(3) $S=1!+2!+3!+\cdots\cdots+n!$

8. 某学习小组9名学生，其学号为1,2,…9，期末考试数学，力学，外语，及格者的名单为:

学　科	及格者的学号
数学	9，4，1，6，3，7，8，
外语	8，1，6，9，4，5，3，7
力学	2，5，9，1，3，7，

打印出三科都及格的学号名单。

第五章 循环结构

§5-1 循环的概念

一、循环的引入

使用计算机解题与人工解题的目标是相同的，都希望以尽量少的工作量，尽量短的时间，获得一定精度的结果。但是它们的思路却有很大差别。人工解题往往偏重于运用人的知识、经验或技巧，选择适当的形式来达到这一目标，而使用计算机工作，则希望采用重复而简单的递推操作，完成复杂问题的求解，以充分发挥计算机速度快、精度高、容量大的优势。为此程序设计中经常采用重复结构。例如：

(1) 累计1＋2＋3＋…＋100，每加一项打印当时的累加值。

(2) 1982年统计我国有十亿三千万人口，假设人口年平均增长率为2%，问从1982年算起多少年后我国的人口增加到十三亿。

对于问题(1) 要反复执行加法运算，共执行100次。

对于问题 (2)，假设Q为某年我国人口数，P为下一年我国人口数，由年平均增长率 R 值可计算出下一年我国人口P为：P＝Q＋Q×R得出新的一年人口P后，判断一下P是否已达到12亿。如未达到，再用上式计算。此时的上式可写成如下形式：

$$P=P+P\times R$$

因此，若一开始就把我国1982年的人口数赋给P，则计算下一年的人口数就是反复计算上式，直到P≥12亿为止。

解决这两个问题需要重复进行特定的运算，在程序设计中，就是要使完成特定的运算的语句序列或程序段重复执行。这种为计算求解需要重复执行指定的语句序列或程序段的结构称为循环结构。

二、循环的结构

对于问题 (1) 可以根据图 5-1 所示的流程图进行程序设计，虚线框内的部分要反复执行100次。

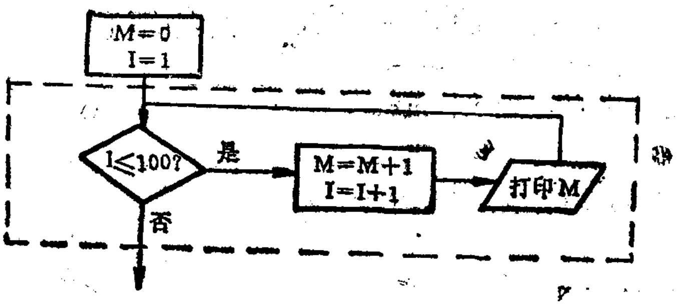

图5-1 计数器控制循环结构流程图

问题 (2) 可以根据图5-2所示的流程图编写程序。

虚线框内的部分要重复执行多次，至于重复执行的次数事先无法知道，何时退出虚线框

内循环部分要根据判断条件作出选择，只有当条件满足时才退出循环结构。

图5-1和图5-2是两种典型的循环结构。图5-1所示的循环结构其循环部分的执行次数事先是知道的，故重复执行循环部分的次数可由一个计数器来控制，这种循环结构我们称为计数器控制循环。而图5-2中的循环部分重复执行的次数不能事先知道，这时就不能用计数器来控制循环，而需要用条件来控制循环，这种循环结构我们称为条件控制循环结构。

图5-2的条件控制循环结构还有另外一种形式，见图5-3所示。

虚线框内是循环部分。执行过程是：当条件“$P<13\times10^8$”满足时反复执行“P＝P＋P×R”；否则（即条件不满足时）就不执行“P＝P＋P×R”。

这两种形式的条件控制循环结构的不同之处在于：(1) 图5-3是先判断条件后执行“P＝P＋P×R”，如果条件不满足则“P＝P＋P×R”一次也不执行；而图5-2是先执行“P＝P＋P×R”再判断条件，“P＝P＋P×R”部分至少执行一次。(2) 图5-3是当条件不满足时结束循环，而图5-2是当条件满足时结束循环。

图5-2所示的循环结构称为“直到型循环结构”，又称UNTIL型结构，因为它是“直到满足条件时就不执行了”，先执行后判断。图6-3所示的循环结构称为“当型循环结构”又称为WHILE型结构，因为它是“当条件满足时执行循环部分”，先判断后执行。

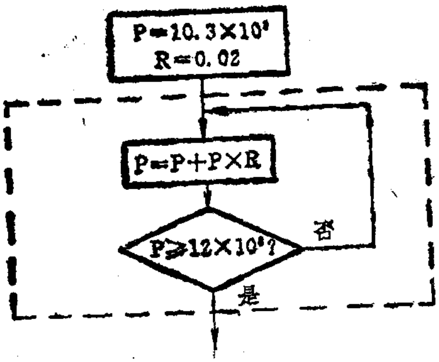

图5-2　直到型循环结构流程图

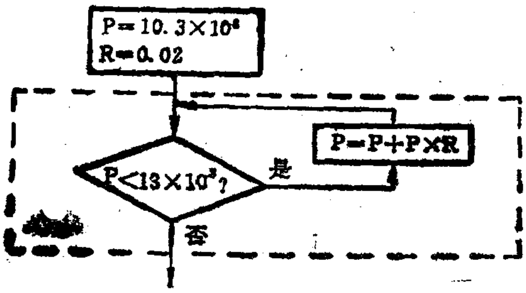

图5-3　当型循环结构流程图

True BASIC提供了两种循环结构：FOR 循环和 DO 循环。前者是一种计数器控制循环；后者是一种条件控制循环，使循环控制更加灵活。True BASIC具备了“直到型循环”和“当型循环”两种形式的条件控制循环，而且循环结构的形式更为丰富，下面将详细介绍。

§5-2　FOR-NEXT 循环结构

一、FOR-NEXT循环的一般格式及执行过程

FOR-NEXT循环的一般格式为：

FOR〈循环变量〉=〈初值〉TO〈终值〉STEP〈步长〉

· · ·
· · · }（循环体）
· · ·

NEXT〈循环变量〉

其中，循环变量是在FOR循环中控制循环次数的变量，只可以是数字型变量；初值、终值可

以是正负整数或实数；步长是指循环体每重复执行一次后，循环变量应变动的数值，可以是正负整数、实数，当步长值为1时，STEP可省略；NEXT中的循环变量名应与FOR语句中的变量名一致。

我们举一个简单的例子来说明FOR循环的执行过程。

计算由三个三角形组成的多边形面积，见图5-4所示，已知：a=3、b=4，c=5，d=6，e=7，f=6，g=8。

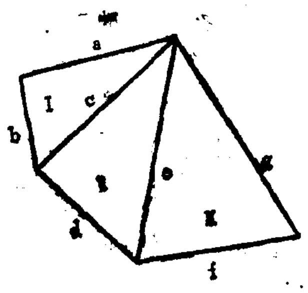

图5-4　多边形面积图

程序及运行结果为：

```
 LET t=0
FOR i=1 TO 3 STEP 1
    REAb a,b,c
    LET P=(a+b+c)/2
    LET S=Sqr(P*(p-a)*(p-b)*(p-c)
    PRINT "s(";i;")=";s,
    LET t=t+s
NEXT i
PRINT "t=" ;t
DATA 3,4,5,6,7,8,5,6,7
END
OK.RUN
s(1)=6      s(2)=20.3332      s(3)=14.6969
t=41.0301
```

其循环过程通过表5-1及流程图5-5加以说明。

概括起来，FOR-NEXT 循环总是先给循环变量赋值，在NEXT语句处对循环变量进行变更（环循变量=循环变量+步长），并检测循环变量取值是否超出终值，若不超过终值，则继续执行循环；若超过终值，则结束循环，执行NEXT以后的语句。使用FOR-NEXT 循环语句应注意以下几点：

(1) 循环变量必须是简单变量；

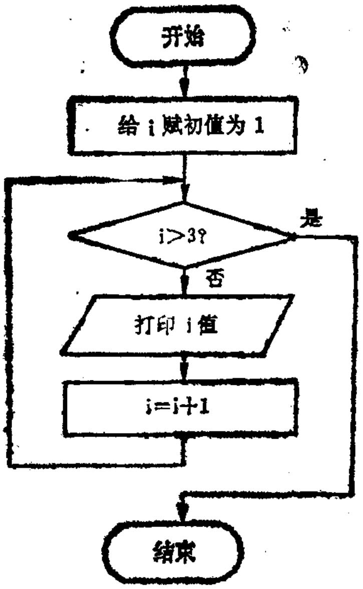

图5-5　循环流程图

表5-1　循环过程表

第几次循环	i值	与终值比较	是否执行循环体
1	1	<3	执行
2	2	<3	执行
3	3	=3	执行
4	4	>3	停止执行

（2） 当终值>初值时，步长应该为正；

（3） 当终值<初值时，步长应该为负；

（4） FOR-NEXT必须成对出现：用它们构成了循环体的起止界限，两者缺一不可，计算机将因无法判定循环体的起止界限，而显示出错信息：

ENDING DOESN'T MATCH BEGINING

二、循环嵌套

在 FOR-NEXT 循环的循环结构体中，可以包括任何其它语句，其中也可包含FOR-NEXT语句和其他结构语句，这种在循环体内部，再使用循环语句的循环称为多重循环（或循环嵌套）。

下面通过一个例子，计算并打印出乘法九九表，说明循环嵌套的结构。

“乘法九九表”的程序流程图见图5-6所示，图中虚线框内部是内层循环。根据图5-6流程图编写的程序为：

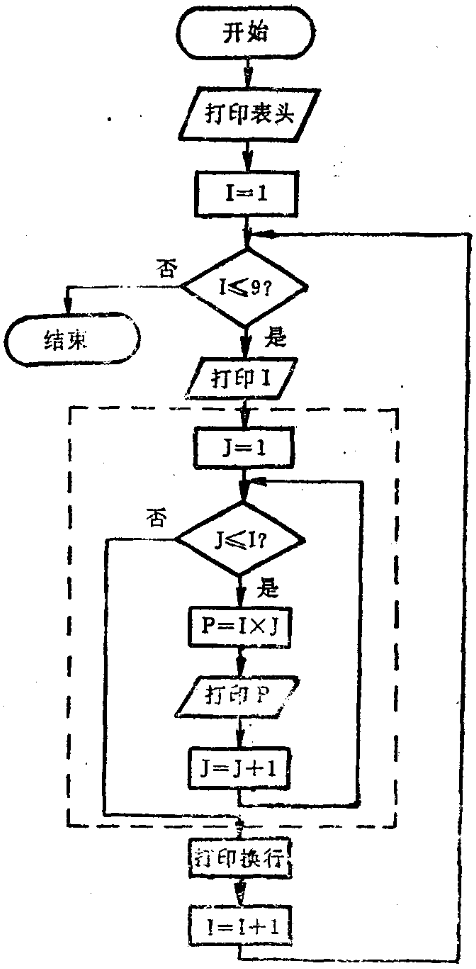

图5-6 “乘法九九表”程序流程图

```
! Print a multiplication table
!
PRINT "*" ;
FOR a=1 TO 9
   PRINT tab(4*a+1);a;
NEXT a
PRINT
FOR i=1 TO 9
   PRINT i;
   FOR j=1 to i
      LET p=i*j
      PRINT tab (4*j+1); p;
   NEXT j
   PRINT
   NEXT i
   END
```

程序中内循环的后面，有一个空输出项的PRINT语句，作用是换行打印。

这个程序执行过程中，i的值是1时，就从1变到1，打印出P＝i＊j＝1×1＝1；然后，i 变为2，j从1变到2，打印出2＊1＝2，2＊2＝4；…。即j的变化次数随i的取值变化而变化。在整个程序执行过程中，外层循环共执行9次，而内层循环共执行了$\sum_{i=1}^{9} i=50$次。

三、EXIT FOR语句

EXIT-FOR语句可以使程序从包含EXIT-FOR语句的最内层FOR循环中退出来，转到NEXT语句后面的程序继续执行，而不必等到循环变量到达终值时。在FOR循环体内，可以根据需要放置任意多个EXITFOR语句，但是要注意，在循环体外面，不可使用这条语句。一个完整的FOR-NEXT控制语句一般格式为：

```
FOR     〈循环变量〉=〈初值〉 TO〈终值〉STEP〈步长〉
        [语句块1]
        [EXIT FOR]
        [语句块2]
NEXT    〈循环变量〉
```

例1 编写一个True BASIC程序，打印输出一个这样的三位数：这个三位数本身恰巧等于其每个数字的立方和（如371＝$3^3+7^3+1^3$）。

寻找的这样的三位数可能有多个，根据题目的要求只要找到一个这样的三位数程序就终止。程序采用了三重循环嵌套，所以一旦找到了满足要求的三位数后就用EXIT FOR语句退出循环。程序及运行结果如下。

```
   !
   !
OR i =1 to
    FOR j=1 to 9
         FOR k=1 TO 9
              LET p=100*i+10*j+k
              LET q=i^3+j^3+k^3
              IF p=q THEN
                 PRINT "x=" ;p
                  EXIT FOR
               END IF
           NEXT k
           IF p=q THEN
              EXIT FOR
           END IF
       NEXT j
       IF p=q THEN
          EXIT FOR
       END IF
  NEXT !
  END
  OK.RUN
   x=153
```

§5-3 DO循环控制结构

DO循环让你重复执行语句组成的循环体，DO 循环与FOR-NEXT循环不同之处在于循环次数并不明确。换句话说，DO循环的重复次数取决于程序运行时逻辑判断结果。

一、最简单的DO循环格式

最简单的DO循环格式为：

```
DO
    · · ·
    · · · } 〈循环体〉
    · · ·
LOOP
```

这种DO循环结构是没有条件的，循环体将无限循环地执行下去。要中止循环的运行，须按键盘上的暂停键Ctrl-Break。

一般来说，循环是有条件的，即用户常常要求程序能在一定的条件下结束循环，这时可使用EXIT DO语句。在DO循环结构的循环体中，当程序执行到EXIT DO语句时，控制立即转到LOOP语句下面的程序段继续执行，EXIT DO语句强制中止了循环。

例2 编写累乘器程序。累乘器程序的作用是能将输入的各数据相乘。

程序及运行结果为：

```
! Program of re-multiply
!
LET m=1
DO
   INPUT PROMPT "x=" :x
   IF x=0 THEN EXIT DO
   LET m=m*x
   PRINT "m=" ;m
LOOP
END

OK.RUN
x=2
m=2
x=3
m=6
x=5
m=30
x=4
m=120
x=0
```

程序中累乘器m的初值应置为1，因为如初使m＝0，则所有累乘的结果就会得零。同理，输入的x也不应该等于零。所以我们规定一旦输入了x＝0，便使程序退出循环，停止累乘，这里“x＝0”，是程序员自己指定的“循环结束标志”。当用户想要结束程序运行时，可给x输入一个“0”。

二、WHILE型和UNTIL型

除了EXIT DO语句外，我们还可以使用WHILE〈条件〉和UNTIL〈条件〉来控制循环。True BASIC提供了多种灵活的循环结构形式，WHILE〈条件〉和 UNTIL〈条件〉可以放在DO语句中，也可以放在LOOP语句中，使用起来十分方便。这样DO循环结构共可组成六种标准格式，其中一般格式为：

格式1：DO UNTIL〈条件〉

```
• • •
• • • }〈循环体〉
• • •

LOOP
```

格式2．DO WHILE〈条件〉

```
• •
• • }〈循环体〉
• •

LOOP
```

格式3：DO

```
• • •
• • • }〈循环体〉
• • •

LOOP UNTIL〈条件〉
```

格式4：DO

```
• • •
• • • }〈循环体〉
• • •

LOOP WHILE〈条件〉
```

格式5：DO WHILE＜条件1＞

```
• • •
• • • }〈循环体〉
• • •

LOOP UNTIL〈条件2〉
```

格式6：DO UNTIL〈条件1〉

```
• • •
• • • }〈循环体〉
• • •
```

LOOP WHILE〈条件2〉

格式1～格式6各自执行的流程图见图5-7(a)～(f)所示：

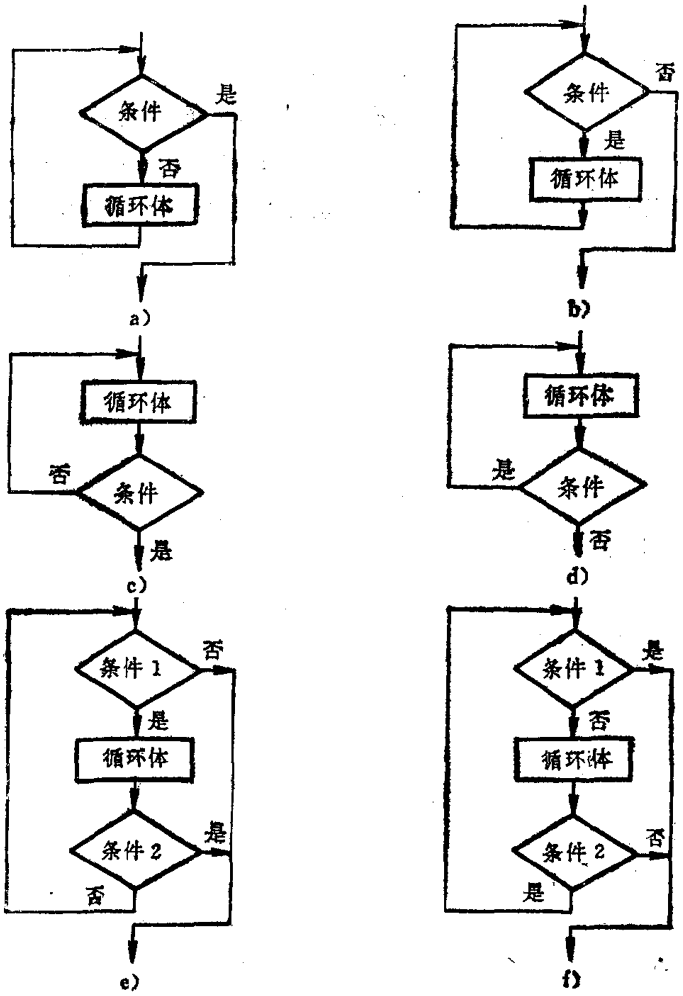
条件
是
否
循环体
a)
条件
否
是
循环体
b)
循环体
否
条件
是
c)
循环体
是
条件
否
d)
条件 1
否
是
循环体
条件 2
是
否
e)
条件 1
是
否
循环体
条件 2
否
是
f)

图5-7　六种DO循环结构形式流程图

请读者注意图5-7各图中判断框出口处“真”和“假”的位置。

三、条件控制环循的使用说明

下面对WHILE型和UNTIL型循环作几点语明：

(1) “条件”可以是任意的逻辑表达式，如DO UNTIL a>b AND c>d。

(2) WHILE型是当条件为真时，循环体被反复执行；UNTIL 型是当条件为真时，就停止执行循环体。因此要把WHILE型改为UNTIL型或UNTIL型改为 WHILE 型，条件要变反。例如，“DO WHILE x<>0”和“DO UNTIL x=0”作用相同。

例3　假定某人在银行存款10000元，银行的年利率为8.5%。那么存入后的第一年结束将获利850元。如果将此利继续存入银行，第二年结束时获利将比850元稍多。如此类推，每年利息都有所增加。试问几年后连本带利会增加一倍？

首先我们采用DO UNTIL型循环。

```
! Compute interest on a blank account.
! Stop when the money has doubled.
!
LET m=10000
LET n=m
LET i=8.5/100
DO UNTIL m>=2*n
   PRINT y,m
   LET y=y+1
   LET m=m+(i*m)
LOOP
PRINT "In" ;y; "years,you'll have" ;m
END
OK.RUN
0                 10000
1                 10850
2                 11772.2
3                 12772.9
4                 13858.6
5                 15036.6
6                 16314.7
7                 17701.4
8                 19206.
In 9 years,you'll have 20838.6
OK.
```

程序中，变量y表示开始存款后的第几年；变量m表示某人第y年后在银行存款的钱数；变量n和i分别表示某人开始在银行存款的钱数和银行的年利率（把n和i看成变量而不是常量是考虑到该程序的通用性，以便以后再使用此程序时更改起来方便）。程序一开始用了三条LET语句分别给变量m，n和j赋初值。

下面DO UNTIL语句的含义是，重复执行下面的循环体直到m的值大于或等于两倍的n值时为止。PRINT语句显示了第y年后某人在银行的存款钱数m。接着一个LET语句用来修改y的值以指向下一年。最后一个LET语句用于计算新的一年连本带利存入银行的钱数，作为m的新值。LOOP语句表示这一次循环运行的结束，并将控制转交DO UNTIL语句。

每次执行DO循环之前总要检查条件：m>=2＊n是否成立。如果条件一开始就成立，系统就一次也不执行这一DO循环了。

当程序运行到y=9时，即第九年结束时，存入银行的钱数m=20838.6时，控制转向DO UNTIL语句，由于这时的m>2＊n。判定条件为真，True BASIC 跳过循环去执行紧跟着LOOP语句的PRINT语句，执行完PRINT语句后遇到END语句，程序即停止运行。

DO UNTIL 循环是当条件为真之前将重复执行的循环。我们还可以使用当条件为真时重复执行的程序，这就是DO WHILE型循环结构。DO WHILE型循环结构和DO UNTIL型循环结构，执行过程基本一样，只是在判断条件上要变反。对前例若采用DO WHILE型循环结构，程序为：

例4

```
! Compute interest on a blank accotun.
! Stop when the moncy has doubled.
!
LET m=10000
LET n=m
LET i=8.5/100
DO WHILF m<2*n
   PRINT y,m
   LET y=y+1
   LET m=m+(i*m)
LOOP
PRINT "In" ;y; "years, you'll have" ;m
END
```

上程序运行结果与采用DO UNTIL 结构的程序运行结果相同。所以，为了获得相同的效果，WHILE〈条件〉中的运算符应该与相应UNTIL〈条件〉中所用的运算符相反。

(3) WHILE〈条件〉或UNTIL〈条件〉放在DO语句中与放在LOOP语句中，程序执行的情形是不相同的。如果把条件放在DO语句中，则在进入循环前先对判断条件进行测试，所以有可能因条件不成立而一次循环也不进行（对WHILE循环而言）；若条件放在LOOP语句中，则程序至少要执行一次循环然后才对条件进行测试，以决定是否要执行一次循环体。

四、控制DO-LOOP循环的二个逻辑函数

1.END DATA和MORE DATA逻辑函数

在循环语句中常使用READ/DATA语句完成数据的输入，为了保证循环中的READ语句读完DATA的数据跳出循环，True BASIC设计了END DATA和MORE DATA二个逻辑函数，它们可保证READ语句一一对应地将DATA数据读入内存。它们与DO-LOOP语句配合使用有表5-2所列形式。

表5-2　DO-LOOP语句的四种使用格式

第一种格式	第二种格式	第三种格式	第四种格式
DO WHILE MORE DATA	DO	DO UNTIL END DATA	DO
循环体	循环体	循环体	循环体
LOOP	LOOP WHILE MORE DATA	LOOP	LOOP UNTIL END DATA

这四种形式都要求循环体内有READ语句。当READ语句对应DATA语句读取数据时，

没有读完，则执行循环体中的语句。若读完数据，则跳出循环体，执行LOOP语句的下面语句。例5为第一种格式。

例5 设我国1980年工业产值为100，按每年增长率c%，求当c分别为6，8，10，12，14时，工业产值几年后可以翻一翻？计算工业产值的公式为：$pl=pl*(1+c\%)^y$。

源程序及运行结果为：

```
PRINT"年增长率c%","年数y", "产值pl"
DO WHILE MORE DATA          !  当数据读完时，跳出循环体
   READ c                   !  读入不同的年平均增长率
   LET p=100                !  1980年产值
   LET pl=p                 !  累积产值的初值
   LET y=0                  !  累加年数的初值
   DO UNTIL pl>=2*p
   LET pl=pl+pl*c/100       !  累加年产值
   LET y=y+1                !  累加年数
   LOOP
   PRINT c,y,pl
LOOP
DATA 6,8,10,12,14
END
OK.run
年增长率 c%           年数 y             产值 pl
   6                  12                 201.22
   8                  10                 215.892
   10                 8                  214.359
   12                 7                  221.068
   14                 6                  219.497
```

表中的其它三种格式，也可以达到同样目的，读者可自己编辑进行练习。

2.KEY INPUT逻辑函数

KEY INPUT逻辑函数在循环中与其它有关语句配合使用。只要键盘输入任何一个键值，DO-LOOP循环就终止运行。例如在下边DO-LOOP程序中使用了KEY INPUT逻辑函数，只要键盘不输入键值，它总是循环运行，在命令窗口显示$x_1=5$，$x_1=10$，…。一旦键入一个任意键值，就立即跳出循环，执行下边的语句。

```
LET X1=0
DO
   LET x=5
   LET x1=x1+x
   PRINT x1;" ";
LOOP UNTIL KEY INPUT
PRINT
END
OK.RUN
5   10   15   20   25   30   35   40   45   50   55   60   65   70   75   80
85  90   95   100  105  110  115  120  125  130  135  140  145  150
```

```
155   160   165   170   175   180   185   190   195   200   205   210   215
220   225   230   235   240   245   250   255   260   265   270   275   280
285   290   295   300   305   310   315   320   325   330   335   340   345
350   355   360   365   370   375   380   385   390   395   400   405   410
415   420   425   430   435   440   445   450   455   460   465   470   475
480   485   490   495   500   505   510   515   520   525   530   535   540
```

KEY INPUT 逻辑函数还可与其他语句配合使用。

§5-4 应用举例

例6 设计一程序利用FOR-NEXT循环求n!

大家知道，n! =1×2×3×4×…×n。利用 FOR 循环可以比较容易地设计出求 n! 的 True BASIC程序，程序及运行结果为：

```
! Count the fact of n
INPUT prompt"please input the n:":n
LET k=1
FOR i=1 TO n
   LET k=k*i
NEXT i
PRINT"The fact of the";n;"is";k
END
OK.RUN
Please input the n:5
The fact of the 5 is 120
```

例7 设计一个程序，使计算机能显示图5-8的图形

分析图形：这个几何图形的行、列都有规律：共显示8行，每行显示的起始位置都比前一行少一格，每行星号数都是奇数，并且从1开始每行递增两个星号。

对于显示这样一个几何图形的问题，程序宜采用双重循环来设计。其中，外循环用来控制打印的起始位置；而内循环则用来控制每一行打印多少个“*”号。

```
              *
             ***
            *****
           *******
          *********
         ***********
        *************
       ***************
```

图5-8 三角图形

显示几何图形的True BASIC序程为：

```
! Display a geometric figure
!
CLEAR                        ! 清屏幕
FOR i=1 TO 8
   PRINT tab(41-i);
   FOR j=i TO 2*i-1
        PRINT "*";
   NEXT j
   PRINT
NEXT i
```

```
END
```

程序利用了 TAB(X) 函数来控制每行打印的起始位置，从图形分析知道，行数i每增加1，打印的起始位置就少1格，所以考虑设x＝N－i，这里N是某个常数。为使图形显示在屏幕中间第40格处，取N＝41。

例8 将一个班的考试成绩放在DATA语句中，计算并打印全班的平均成绩及最高、最低分数。设数据语句为 DATA 71，62，100，70，93，77，63，80，87，79，85，90。

程序及运行结果为：

```
∫ Count an examination report
!
LET n=0
LET min=100
LET max=0
LET s=0
DO WHILE MOHE DATA
    READ cj
    LET s=s+cj
    IF cj>max THEN LET.max=cj
    IF cj<min THEN LET min=cj
    LET n=n+1
LOOP
LET p=s/n
PRINT "The mean of number is" ;p
PRINT "The highest marks is" ;max
RPINT "The lowest marks is" ;min
DATA 71,62,100;70,93,77,63,80,87,79,85,90
END
OK.RUN
The mean of number is 79.75
The highCst marks is 100
The lowest marks is 62
OK.
```

程序中变量n和s分别表示统计全班人数和总分；min和max分别表示最低分数和最高分数，它们的初值分别赋予100和0，因为全班成绩的最低分不会大于100，而最高分不会低于0分。程序中DO循环条件采用了“MORE DATA”测试，这是用来检查是否还有数据要读入当还有更多的数据项未读入时，则继续循环，直到读完数据为止。

在 §5-3，四中，我们介绍了“END DATA”测试。MORE DATA和END DATA测试都是用来检查是否还有数据要读入，凡是逻辑表达式能够使用的地方，END DATA 和 MORE DATA都能出现，而且还可以与其它逻辑条件组合起来使用。这两种测试的逻辑值正好相反：当程序还有数据未读时，MORE DATA为真，END DATA 为假；否则当数据读完之后，END DATA为真，而MORE DATA为假。MORE DATA和END DATA测试是程序设计中一种很有用的方法，它使得在DATA语句中改变数据的总数变得容易得多。程序员可以不必去计算数据项的个数或数据项个数的变化值。

例9 求两个正整数m，n的最大公约数。

在数学中求最大公约数的方法是采用欧几里德算法，也就是对m，n 用辗转相除法，直到余数为零时为止，最后一个除数即为最大公约数。图5-8是求两个正整数m，n的最大公约数的程序流程图。

根据图5-9设计的程序及运行结果为：

```
! Program of Euclid
!
! This program calculates the greatest common
! divisor of two integer numbers using euclidean
! algorithm
PRINT "Get two integer numbers:" ;
INPUT m,n
DO
   LET r=mod(m,n)
   LET m=n
   LET n=r
LOOP UNTIL r=0
PRINT "The greatest common divisor is:"
PRINT m
END
OK.RUN
Get two integer numbers: 18,30
The greatest common divisor is:6
OK.
```

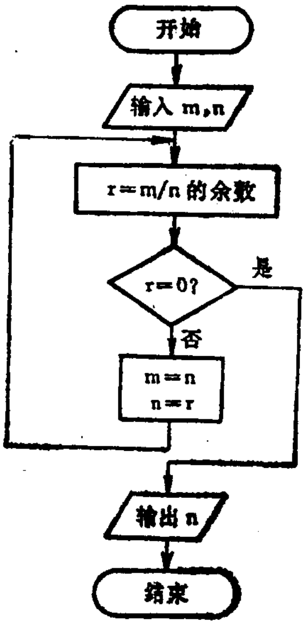

图5-9求最大公约数的程序流程

在上面程序中，如果我们把UNTIL〈条件〉放在DO语句中，情况就不同了。改动的程序为：

```
! Program of Euclid
!
! This program calculates the greatest common
! divisor of two integer numbers using euclidean
! algorithm
PRINT "Get two integer numbers:" ;
INPUT m,n
DO
   LET r=mod(m,n)
   LET m=n
   LET n=r
LOOP UNTIL r=0
PRINT "The greatest common divisor is:" ;
PRINT m
END
OK.RUN
Get two integer numbers:12,30
The greatest common divisor is:12
OK.
OK.RUN
```

```
Get two integer numbers:56,72
The greatest common divisor is:56
```

我们很容易计算出12和30这两个正整数的最大公约数是6；56和72的最大公约数是8。将这两组数输入以上程序。程序运行的结果为：

显然，这个改动的程序有错。原因是“UNTIL r=0”是放在DO语句中的，所以要先测试。而程序事先未对r赋过值，True BASIC 在程序运行前使全部变量的初值为零，因此一开始测试条件r=0就满足，循环体根本不被执行。解决的方法是，应在DO循环之前给r赋一个适当的值，以便进入循环。经过调整过的 True BASIC 程序清单及程序运行结果为：

```
! Program of Euclid
!
! This program calculates the greatest common
! divisor of two integer numbers using euclidean
! algorithm
PRINT "Get two integer numbers:" ;
INPUT m,n
LET r=mod(m,n)
DO UNTIL r=0
   LET m=n
   LET n=r
   LET r=mod(m,n)
LOOP
PRINT "The greatest common divisor is:" -
PRINT n
END
OK.run
Get two integer numbers:21,56
The greatest common divisor is:7
OK.
OK.run
Get two integer numbers:15,25
The greatest common divisor is:5
OK.
OK.run
Get two integer numbers:10,5
The greatest common divisor is:5
```

习 题 五

1有哪几种典型的循环结构形式？分别举一例子，并编写True BASIC程序。

2.FOR循环的控制特点是什么？设计一个程序，求1～9的平方、立方。

3.DO循环控制的特点是什么？

4.写出下列程序的执行结果。

```
(1) FOR y=-5 TO -9 STEP-1
        LET x=y+1
        PRINT x,

    NFXT y
    END
```

```
(2) LET x=2
    FOR t=5 TO 10 STEP 2
        LET p=t*x
         PRINT p,
    NEXT t
     END
```

```
(3) FOR x=5.5 TO 9 STEP 1.5
        PRINT x,
    NEXT x
    END
```

```
(4) FQR i=1 TO 4
        FOR   j=1 TO i
          PRINT INT((i+1)/2);
        NEXT j
     PRINT
     NEXT i
     END
```

```
(5) DO
        LET n=n+1
        LET s=s+2 n
    LOOP UNTIL s>100
    PRINT n,s
    END
```

```
(6) LET i=1
    LET n=1
    DO WHILE  n<=15
        PRINT "i=" ;i, "n=" ;n
        LET i=i+1
        LET n=n+1
    LOOP
    PRINT"m=";i-1
    END
```

5. 设计一个程序要求用户由键盘输入一系列正数，当输入负数时程序停止处理，并显示出输入正数的个数，所有正数的总和及平均值。

6. 画流程图，并写出求n的最大值的程序。

$S=1+2^2+3^2+\cdots\cdots+n^2\leqslant 10000$

7. 一个数列，它的头两个数为0和1，第三个数是第一、二个数之和，第四个数是第二、三个数之和，以后每个数都是其前两个数之和，请编写程序，计算并打印出这个数列，直到第60个数为止。

8. 编写用＊打印下列各图形的程序（见图5-10，图5-11，图5-12）。

(1)

```
* * * * * * * *
 * * * * * * * *
  * * * * * * * *
   * * * * * * * *
    * * * * * * * *
     * * * * * * * *
      * * * * * * * *
       * * * * * * * *
```

图 5-10

(2)

```
             *
            * *
          *     *
        *         *
      *             *
    *                 *
  *                     *
*                         *
* * * * * * * * * * * * * *
```

图 5-11

(3)

```
        0
      1 1 1
    2 2 2 2 2
  3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4
  5 5 5 5 5 5 5
    6 6 6 6 6
      7 7 7
        8
```

图 5-12

9. 编写一个程序，计算并打印出下列数据中的最大值与第二大值。

3,15,6,26,-3,-9,35,22,91,77

第六章　数组与矩阵

§6-1　数组的基本概念

一、数组的引入

迄今为止，我们使用的变量都是简单变量，其特点是一个变量名对应一个变量，每个变量名之间没有任何联系。但是这种变量的命名方法在一些程序中，特别是在大量有联系的数据方面，有时是很困难的，甚至是不可能的。

在现实世界中碰到的事物之间往往是有联系的。例如：把读入的一百个数按照读入顺序的逆序打印出来，要求解决这个问题，必须首先把一百个数读入并存放到不同的变量中，然后才能按要求打印。如果我们用已学过的简单变量来存储这些数据，例如用A、B、C…分别存储顺序读入的一百个数，这些简单变量不仅不直观，而且不能反映数据的读入顺序。这样，在程序设计中按要求逆序输出读入的数据，需要将存放的数据的一百个变量按读入的逆序全部列出来，这给程序设计带来了很大的麻烦。

为了提高编程效率，人们把这些有联系的数据组合在一起成为一个数组。数组不是代表一个变量，而是代表有内在联系的一批变量，数组内的每个变量称为该数组的元素。我们希望从一个变量名就既能知道该变量属于哪个数组，又能知道它是该数组中的第几个变量，从而加以区分。如统计四名学生的平均成绩。成绩表见表6-1所示。

表　6-1

姓　　名	英　　文	数　　学	物　　理	计算机
李云	87	89	67	88
黄中	66	77	88	99
王林	88	93	100	87
孙吴	7	66	98	100

若用简单变量表示姓名需要四个，成绩需要16个，而且反映不出它们之间的联系。若用数组表示则一目了然。

现将学生姓名用数组name$表示，学生成绩用数组a表示。其中数组name$由4个下标变量组成：

name$(1)	存放李云
name$(2)	存放黄中
name$(3)	存放王林
name$(4)	存放孙吴

数组a由16个下标变量组成：

a(1,1)	a(1,2)	a(1,3)	a(1,4)
a(2,1)	a(2,2)	a(2,3)	a(2,4)
a(3,1)	a(3,2)	a(3,3)	a(3,4)
a(4,1)	a(4,2)	a(4,3)	a(4,4)

分别存放四个学生四门课程成绩。

程序及运行结果为：

```
DIM name$ (4) , a (4, 4)          ! 定义数组
FOR i=1 TO 4
    READ a$
LET name$(i)=a$
    FOR i=1 TO 4
        READ num
        LET a(i,j)=num
        LET total=total+num
    NEXT j
! 打印每个学生平均学科成绩
PRINT name $(i); "四科平均成绩为" : ; total /4
NEXT i
DATA "李云" ,87,89,67,88, "黄中" ,66,77,88,99
DATA "王林" , 88,93,100,87, "孙吴" ,78,66,98,100
END
OK.RUN
李云    四科平均成绩为: 82.75
黄中    四科平均成绩为: 82.50
王林    四科平均成绩为: 92.00
孙吴    四科平均成绩为: 85.50
OK.
```

执行程序时，通过圆括号中数字的变化可以找到不同人名和不同学科成绩存储单元，就象前面用过的A，B，C等变量一样，可以对name $(i) 及a(i, j) 单元存取数据。i和j的值可以在一个很大的范围内变化，于是name$(i)和a(i, j) 也相应地代表了许多个变量。

二、数组的定义

数组是按一定顺序排列且具有相同性质的变量的集合，它与集合不同的是其有序性。一个数组用一个名字来命名，其中每个元素代表一个变量。数组元素是用数组名与元素序号（或称下标）来共同确定的。

数组在使用之前，必须预先对数组进行定义：确定数组名及数组大小及维数。数组命名规则与变量名相同，并且也分为数字型与字符串型两类。数组定义使用DIM(DIMENSION)语句，其格式为：

DIM 〈数组名表〉

其中，〈数组名表〉代表1到多个数组，每个数组之间用逗号分开。

例 DIM A(5), B$(3, 4)

共定义了两个数组，其中A(5) 定义了一个一维数值型数组，数组名为A，共有5个元

素；B$(3，4) 定义了一个名为B$的二维字符串数组，共有3行4列 12 个元素，见图 6-1 所示。

数组定义语句提供了如下信息：

(1) 数组名及数组类型。

(2) 数组维数。

(3) 各维的大小。

(4) 下标的上下界。

数组定义中应注意以下要求及特点：

A(5)					
	A(1)	A(2)	A(3)	A(4)	A(5)

B$(3.4)				
	B$(1,1)	B$(1,2)	B$(1,3)	B$(1,4)
	B$(2,1)	B$(2,2)	B$(2,3)	B$(2,4)
	B$(3,1)	B$(3,2)	B$(3,3)	B$(3.4)

图6-1　数组元素数据结构示意图

数组定义后一般不允许重复定义，故数组定义语句通常放在程序开头；数组定义使用的数组名在True BASIC中不得再用于简单变量名；数组元素在定义后未经赋值前默认有以下取值：数值型数组各元素初值均为0，字符串数组各元素初值均为空；在数组定义中，括号定义维数及数组大小时，必须使用常数，不可使用变量。

一维数组相当于一个一维表，二维数组相当于一个二维表。多维数组视机型而定，PC机可最多定为255维。数组中每个元素名用数组名及括号中的下标值来表示。例如A(5)有五个元素，分别用 A(1)，A(2)，A(3)，A(4)，A(5) 来表示。B$(1，1)，B$(1，2)……代表二维数组 B$(3，4) 中的各元素，请参看图 7-1。B$数组各元素在未赋值前均为空。

三、数组定义中的上下界

在使用数组定义语句DIM定义数组时，实际上已确定了下标的上下界，所谓下标上、下界就是数组下标可以使用的最大、最小值。所定义数组的大小，可以超过实际使用的元素数，但不得小于使用元素数。

例如　DIM A(5) 定义了A数组有五个元素，元素的上界是5，下界是1。

在DIM语句中，为了说明数组的上下界，可在括号内写成 (〈下界〉TO〈上界〉)，也可写成 (〈下界〉:〈上界〉) 。上下界值的选择可根据实际问题的需要进行，但〈上界〉与〈下界〉必须是常数。

例如　DIM year(1980 to 1989)，T(−40:40)定义了两个一维数值型数组，第一个数组year的下界为1980，上界为1989，共有10个元素，第二个数组T(−40:40) 定义下界为−40，上界为40，共有81个元素。下界不加说明时为1，也可用 OPTION BASE constant 语句统一说明下界。其中 constant 为任意整数常数，规定为所有未经声明的数组　界为constant所给出的值。

例如在DIM语句前若有

```
OPTION BASE 2
```

语句时，则DIM A(5)，B$(3，4)所定义的各维的下界均为2。若在数组定义中有下界定义时，如DIM C(3 to 9)，则其下界取自身定义值。

四、数组元素的赋值与使用

数组整体来说是一种特殊的数据结构，但其中每一个元素均可作为一个变量来使用，这个元素变量可以象简单变量一样地进行赋值（使用LET，READ，INPUT 语句），一样地引用于表达式中。例1说明了一维数组元素的各种赋值方法。

例1

```
SET MODE "hires"
DIM a(5),b$(5)
LET a(1)=5
LET a(2)=6
LET a(3)=2*a(2)
READ a(4)
INPUT a(5)
PRINT a(1),a(2),a(3),a(4),a(5)
FOR i=1 TO 5
   READ b$(i)
NEXT i
DATA 7,张民, 李新, 刘宜, 王浩
DATA 胡华
END
```

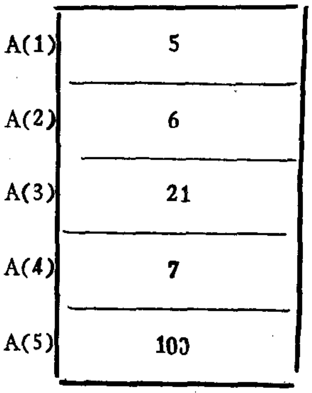

图 6-2

由上例可知，数组元素可象简单变量一样地赋值或使用于表达式中。一旦数组赋值后，其值一直保存于数组中，直到重新赋值；在引用数组元素时，只读出相应数值，并不破坏或改变其数值。图6-2表示数组赋值后数组中数据的存储结构。

§6-2 一维数组及其应用

一、一维数组

数组是True BASIC中一个基本的、十分重要的数据结构，应用十分广泛，从前面例子中，已初步体会到，处理带有顺序关系的数据集合时，采用数组使编程大为直观和简化。

一维数组是由若干个具有相同名称而下标值不同的元素所组成的，它是一个一维表，可根据下标值确定表中的元素。下面通过编写一个记录学生姓名及其成绩的程序，介绍一维数组的应用。

如果要登记30个学生或更多学生的成绩，使用简单变量编写程序将会非常繁琐，因此，在处理大量同类数据时，一般均采用数组数据结构。下面应用数组改写上例程序。

例2

```
SET MODE "hires"
READ xm1$,cj1
READ xm2$,cj2
READ xm3$,cj3
READ xm4$,cj4
READ xm5$,cj5
PRINT "学生姓名及成绩如下: "
PRINT xm1$,xm2$,xm3$,xm4$,xm5$
PRIINT cj1,cj2,cj3,cj4,cj5
DATA 陈小波, 86, 王炳芳, 78, 张  志, 95
DATA 李  林, 60, 高仁英, 90
END
```

例3

```
SET MODE "hires"
DIM xm$(5),cj(5)
FOR i=1 to 5
    READ xm$(i),cj(i)
NEXT i
PRINT "学生姓名及成绩如下: "
FOR i=1 TO 5
    PRINT xm$(i),cj(i)
NEXT i
DATA 陈小波, 86, 王炳芳, 78, 张志, 95
DATA 李  林,60,高仁英,90
END
```

上述程序使用了两个一维数组XM$和CJ。语句DIM说明了这两个一维数组并设置了它们所能包含的元素个数（下界默认为1）。数组XM$是字符串数组，而CJ的类型是数值。字符串数组名应后缀一个美元号$，它们的命名与字符串变量名的命名是相同的。

二、一维数组的应用

数组是一个有序表，给出数组名及其中任一元素的下标值，便可以唯一确定该数组元素的取值。所以数组经常用于存放同类有关数据，用于信息的管理、查询与成批数据处理。在计算机中，数组的数据结构与数学中的矩阵的形式相一致，因此，数组又可用于矩阵的运算（有关矩阵处理详情见 §6-4）。

1.顺序检索

同类数据的检索与管理经常使用数组。例如，要管理一个班学生的成绩，可将学生姓名、籍贯、年龄、成绩分别放入数组，然后可以打印表格及按姓名进行检索。

设存放姓名、籍贯、年龄、成绩的数组名分别为 XM $，JG$，NL，CJ 使用 Read—Data语句读入数据。

例4

```
SET MODE "hires"
DIM xm$(30),jg$(30),nl(30),cj(30)
DO WHILE MORE DATA
    LET n=n+1
    READ xm$(n),jg$(n),nl(n),cj(n)
LOOP
```

```
PRINT “学生姓名及成绩如下: ”
FOR i=1 TO 5
    PRINT xm$(i),jg$(i),nl(i),cj(i)
NEXT i
DATA 陈小波, 北京, 19, 86, 王炳芳, 山东, 20, 78; 张志, 江苏, 18, 95
DATA 李林, 河南, 21, 60, 高仁英, 山西, 24, 90, 刘欢, 辽宁,20,70
! 查找
INPUT Prompt “输入查找姓名” :”:N$
FOR i=1 TO n
     IF N$=xm$(i) THEN
        LET  k=i
        LET find$= “True”
        EXIT FOR
    END IF
NEXT i
IF find$= “True” THEN
   PRINT xm$(k),jg$(k),nl(k),cj(k)
ELSE
   PRINT “查无此人! ”
END IF
END
```

由上例可得出以下几点结论：

(1) 数组定义语句定义的数组大小可以大于实际使用的大小。但余量不可太大，否则存储空间将有较大浪费，但数组大小不得小于实际使用的大小。

(2) 用DO WHILE more data语句读入数据的方式控制灵活，数据可多可少，自动烑完为止。循环中的计数器n用于统计读入多少数据。

(3) 数据一旦读入数组，就可以在任何地方使用数组中的数据，打印列表都非常简单。

(4) 在一个数组中查找数据时，是用输入的数据（查找项）与数组中每一个元素依次比较，若相等，记下元素的下标并设置一个查找标志（Find$），便可退出循环。

(5) 然后根据查到标志是否为“true”确定打印结果。

本例所用的检索方法，称顺序比较法，在数据量不大时，可使用这种方法，它的平均查找次数为n/2次。

2.排序

在上例中查找一个人的姓名是逐个比较的，这是因为数组中的姓名无任何次序，只好一个个地比较。在数据量较大时，这种查找方法要耗费较多的时间。为了提高检索效率，通常要求数据是有序的，即数组中的数据要由大到小，或由小到大排列。我们将数组中的元素按一定次序（降序或升序）排列的操作称为排序或分类（sort）。

下面介绍几种常用的排序方法。

(1) 选择排序法（selected sort）

选择排序法的算法要点是依次在数组中取出一个元素，并依次与其后元素比较，并根据大小进行必要的交换。在一轮比较与交换完毕后，第一个元素必定是该数列中最大（降序）或最小（升序）元素。依此类推，第二个元素是数组中次大或次小元素，直到第n－1个元素

比较并进行必要交换之后，数组中的元素就按大小顺序排序好了。

例5

```
DIM a(10)
DO WHILE MORE DATA
   LET n=n+1
   READ a(n)
LOOP
FOR i=1 TO n-1
    FOR j=i+1 TO n
         IF a(j)>a(i) THEN
            LET t=a(j)
            LET a(i)=a(j)
            LET a(j)=t
         END IF
    NEXT j
NEXT i
DATA 86,90,80,79,81,70
MAT PRINT a
END
```

例6

```
DIM a(10)
DO WHILE MORE DATA
   LET n=n+1
   READ a(n)
LOOP
FOR i=1 TO n-1
     LET  k=i
     FOR j=i+1 TO n
         IF a(j)>a(k) THEN LET k=j
     NEXT j
     IF i< >k THEN
        LET t=a(i)
        LET a(i)=a(k)
        LET a(k)=t
     END IF
PRINT a(i)
NEXT i
DATA 86,90,80,79,81,70
END
```

上例中排序后的数据是由大到小排列的，称为降序，若将A(J)>A(I) 关系表达式换成A(J)<A(I) 时，则结果是由小到大排序，称为升序。

评价一个排序算法的优劣，通常是以其排序速度，占用空间大小等因素来衡量的。重点是速度。排序中花费时间的多少取决于两种操作。

①比较操作（比较次数多少）。

②交换操作（交换次数多少）。

例5中，排序需要$\frac{n}{2}(n-1)$次比较。比较一次中可能交换次数最多为$n-1$次，最少为0次，平均为$(n-1)/2$次，共比较$n(n-1)/2$趟，故总共平均交换$n(n-1)\cdot(n-1)/4$次。

为了减少交换次数，可以对例5程序稍作修改，其程序见例6所示。

例6中设置一个指针K，每轮中仅记下最大一个元素的下标值，仅交换一次，这样排序中平均最多交换$n-1$次，最少交换0次。因而提高了排序效率（因为一次交换操作要三个赋值语句，比一次比较要花费较多时间）。

(2) 冒泡排序法（bubble sort）

冒泡排序法又叫交换排序法，其要点是每次比较两相邻元素，若不满足顺序要求则两者交换，否则不交换，这样经过一轮比较后，一个最大（或最小）元素浮到了数组的顶部，故称为冒泡排序。

例7

```
! bubble sort
DIM a(5)
LET n=0
DO WHILE MORE DATA
    LET n=n+1
    READ a(n)
LOOP
FOR i=n-1 TO 1 STEP-1
     FOR j=1 TO i
          IF a(j)>a(j+1) THEN
             LET t=a(j)
             LET a(j)=a(j+1)
             LET a(j+1)=t
          END IF
    NEXT j
MAT PRINT A
NEXT i
DATA 86, 90, 79, 81, 70
END
```

冒泡排序中比较的次数为$\frac{n}{2}(n-1)$，一轮中交换的次数最大为$n-1$次，最少为0次，总共平均交换次数为$\frac{n}{2}(n-1)$。

排序的算法很多，这里限于篇幅不能一一列举，可参考有关算法的专业书籍。

3.对分检索

数据排序的目的是为了加快检索速度，一个有序的数组在查找某个数据时，就不必要顺序一一比较每一数组元素了。采用对分检索可以大大提高检索效率。

对分检索法又叫二分检索法，其检索原理是将要检索的数据首先与数组中间的元素相比较（设已按升序排列），如果前者较大，则要检索的数据必在数组的后半部中，可再与后半部元素的中间元素比较，依次比较检索，每次可将检索范围缩小一半，重复上述对分检索过程，结果可能有两种：查到要检索的元素；或查不到该元素。

下面通过一个例子说明对分检索的应用。

例8 将一组学生数据按姓名排序、并用对分检索查找。

```
!
SET MODE "egahires"
DIM xm$ (30) , jg$ (30) , nl (30) , cj (30)
DO WHILE MORE DATA
   LET n=n+1
   READ xm$ (n) , jg$ (n) , nl (n) , cj (n)
LOOP
PRINT"学生姓名及成绩如下: "
FOR i=1 TO n
    PRINT xm$(i),jg$(i),nl(i),cj(i)
NEXT i
DATA 陈小波，北京，19,86，王炳芳，山东，20，78，张志，江苏，18，95
DATA 李林，河南，21，60，高仁英，山西，24，90，刘欣，辽宁，20，70
FOR i=1 TO n-1
    LET k=i
    FOR j=i+1 TO n
        IF xm$(j)>xm$(k) THEN LET k=j
    NEXT j
    IF i < >k THEN
      LET t$=xm$(i)
      LET xm$(i)=xm$(k)
      LET xm$(k)=t$
      LET t$=jg$(i)
      LET jg$(i)=jg$(k)
      LET jg$(k)=t$
      LET t=nl(i)
      LET nl(i)=nl(k)
      LET nl(k)=t
      LET t=cj(i)
      LET cj(i)=cj(k)
      LET cj(k)=t
  END IF
NEXT i
! 查找
INPUT Prompt"输入查找姓名: ":N$
LET l=1
LET h=n
DO
```

```
    LET m=(l+h)/2
    IF N$=xm$ (m) THEN
       LET k=m
       LET find$=“True”
       EXIT DO
    ELSE IF xm$(m)>n$ THEN
       LET l=m+1
    ELSE
       LET h=m-1
    END IF
LOOP UNTIL m=l OR m=h
IF find$=“True” THEN
   PRINT xm$(k),jg$(k),nl(k),cj(k)
ELSE
   PRINT“查无此人!”
END IF
END
```

§6-3 二维数组及其应用

一、二维数组

一维数组是一个一维表，故只用一个数字作数组元素下标，二维数组是一个二维表，为表示一个二维数组元素要用两个数字作为下标，两个下标之间以逗号分隔。例如A(8，10) 就是二维数组元素的形式，括号中第一个数字是第一维的下标，第二个数字是第二维的下标。与一维数组一样，除了有数值型的二维数组，还有字符串型的二维数组。例如 AB$(3，4) 。

二维数组的各维下标同样有各自的上界与下界，上界与下界的标识方式与一维数组规则相同。如果省略下界，则默认为1。

例如语句

DIM year(1993 to 1999), observer$(1 to 10, -40 to 40)

DIM语句定义了两个数组：year和observer$，其中year是一个一维数值型数组；observer$是一个二维字符型数组，有10行，81列，共可存储81×10=810个字符型数据元素。

例如，设A数组第一维的上界为8，下界为5；第二维的上界为12、下界为3，则 DIM 语句中应写成：

DIM A(5:8, 3:12)

同样也可用OPTION BASE 语句来确定各维下标的下界。除了二维数组还有三维、四维……等多维数组。

二、二维数组的应用

例9 库房某页记帐见表6-2所示。

计算各种备件总金额，并与品名、数量、单价一起列出，把品名装入name$数组。

程序及运行结果为：

```
DIM a(4, 3), name$(4)
PRINT TAB(32);“售价表”
```

表6-2 备件表

品　　名	数　　量	单　　价
大　　轴	20	120.45
电　　机	6	598.55
垫　　板	147	56.00
油　　缸	19	102.88

```
PRINT TAB(28);"        "==========="
PRINT "______________________________"
PRINT"品名"; TAB(20);"数量"; TAB(40);"单价" ;TAB(60);"金额"
PRINT "______________________________"
FOR i=1 TO 4
    READ name$(i)
    PRINT"    ";name$(i);
    FOR j=1 TO 2
        READ a(i,j)
        PRINT TAB(20*j);a(i,j);
    NEXT j
    LET a(i,3)=a(i,1)*a(i,2)
    PRINT TAB(60);a(i,3)
NEXT i
DATA 大轴,20,120.45,垫板,147,56.00,油缸,19,102.80,电机,6,598.55
END
OK.RUN
```

```
                              售价表
─────────────────────────────────────────────────────────────
              品名         数量          单价          金额
─────────────────────────────────────────────────────────────
              大轴          20          120.45         2409
              垫板         147           56            8232
              油缸          19          102.8          1953.2
              电机           6          598.55         3591.3
```

例10 计算C=A+B，其中 $A=\begin{pmatrix}1 & 2 & 3\\ 7 & 8 & 9\end{pmatrix}$ $B=\begin{pmatrix}4 & 5 & 6\\ 10 & 11 & 12\end{pmatrix}$

程序及运行结果为：

```
! LSUE
DIM a(2,3),b(2,3),c(2,3)
PRINT TAB(15);"a"; TAB(45);"b"
PRINT "________________________"
```

```
FOR i=1 TO 2
   FOR j=1 TO 3
        READ a(i,j)
        PRINT TAB(10*(j+1)-9);a(i,j);
   NEXT j
   FOR j=1 TO 3
      READ b(i,j)
      PRINT TAB(10*(j-1)+40);b(i,j);
   NEXT j
   PRINT
NEXT i
PRINT "________________________"
PRINT TAB(30);"c"
PRINT TAB(15);"________________________"
FOR i=1 TO 2
    FOR j=1 TO 3
    LET c(i,j)=a(i,j)+b(i,j)
    PRINT TAB(10*(j-1)+15);c(i,j);
    NEXT j
    PRINT
NEXT i
DATA 1,2,3,4,5,6,7,8,9,10,11,12
END
OK.RUN
```

```
              a                              b
         ____________________________________________
             1 2 3                         4  5  6
             7 8 9                        10 11 12
         ____________________________________________
                           c
                    ______________
                        5  7  9
                       17 19 21
                    ______________
```

三、数组函数

为实际编程的需要，True BASIC提供了一组数组函数，用以对数组的大小及指定维的上、下界进行测试。这些函数都返回一个数值。因而，它们可以使用在任何数值表达式中。

(1) SiZe (A) 函数值是数组A中当前的元素。

(2) SiZe (A, n) 函数值是数组A中第n维的当前元素个数。

(3) Lbound(A, n) 函数值是数组A的第n维的下界。若省略n及前面的“，”时，表示求一维数组A的下界。

(4) Ubound(A, n) 函数值是数组A的第n维的上界。若省略n及前面的“，”时，表示求一维数组A的上界。

§6-4 MAT 语句

从前面几节的学习可知，对数组中一组数组元素进行数据的存取，是逐个进行传送的。这种逐个传送数组元素数据方法不方便，需要程序员十分仔细逐一处理数组中各个元素，稍有不慎就会遗漏或出错。

实际上，在很多情况下，我们需要对数组进行整体操作。例如二维数组常可以用来表示矩阵，矩阵在常微分方程的数值解和线性方程组的求解等方面有广泛的应用，前面例10中，我们例举了a和b两个矩阵相加的运算。在那里我们是通过对每个元素的处理，将整个二维数组扫描一遍，利用双重循环实现的。

为了提高矩阵处理及运算的效率，True BASIC提供了一组MAT语句(MAT是英文矩阵一词MATRIX的缩写)，程序员能够用一个MAT语句进行一个数组的一次整体操作。这时用不着为了对一个数组的每个元素进行同一处理而建立循环，而只需用一个单独的MAT语句即可。

一、MAT输入、输出语句

前面学过的输入（INPUT、READ）、输出（PRINT）语句都是以简单变量为处理对象的，而MAT输入、输出语句则是以数组为处理对象，可用一个语句完成对数组全部元素的输入、输出。

1.MAT READ语句

格式： MAT READ〈数组名表〉

功能： 将DATA语句所提供的数据，按数组顺序送到MAT READ语句所列的数组中。数组名表由一个或多个数组名组成，多个数组名之间要用逗号“，”隔开。

例11

```
!
DIM a$(5),b(2,3)
MAT READ a$,b
DATA A,B,C,D,E
DATA 1,2,3,4,5,6
MAT PRINT a$,b
END
```

其中 MAT READ A$,B 相当于

```
FOR i=1 TO 5
    READ a$(i)
NEXT i
FOR i=1 TO 2
    FOR j=1 TO 3
        READ b(i,j)
    NEXT j
NEXT i
```

注意：

(1) MAT READ语句中所列的数组名，均应在此语句前由DIM语句说明。

(2) DATA语句中数据的类型、个数、顺序与MAT READ语句中数组名的类型、数组元素个数相对应。

2.MAT PRINT 语句

格式：MAT PRINT〈数组名表〉

功能：依次输出语句中所列数组名的全部数组元素值。例如：

```
DIM A(5)
MAT PRINT A
```

功能相当于

```
FOR I=1 TO 5
PRINT A(I),
NEXT I
PRINT
```

一个MAT PRINT语句可以输出多个数组。依照语句中所列数组名的顺序，输出完一个数组的各个元素值后，自动换行再输出下一个数组各元素的值，见例12所示。运行时，将在第一行显示各城市名称，第二行显示各市人口(以万人为单位)。若要输出的是二维数组，则按行的顺序输出，即输出一行数组元素后，自动换行再输出下一行，见例13所示。

例12

```
SET MODE "hires"
DIM a$(5),b(5)
MAT READ a$,b
PRINT "各市人口数"
MAT PRINT a$,b
DATA "北京", "上海", "广州", "大连","合肥"
DATA 1000,1200,800,300,60
END
```

例13

```
DIM a(3,4)
FOR i=1 TO 3
    FOR j=1 TO 4
        LET a(i,j)=10*i+j
    NEXT j
NEXT i
MAT PRINT a
END
```

3.MAT INPUT语句

格式：MAT INPUT〈数组名表〉

或　　MAT INPUT PROMPT expr$：〈数组名表〉

功能：由键盘输入数组名表中所列数组相应的数据。

每个数组的数据原则上应在一行内送完。若数据多，一行容纳不下时，可以在最后一个数据后面加逗号，表示继续输入，则系统在下一行显示“？”等待键入其余的数据，若未加逗号，而数组相应的数据又未送足时，系统会提示：

```
Too few input items, please add more
```

并在下一行显示“？”等待继续键入数据。

当一个数组的数据输入完成时，在最后一个数据后面加回车，表示结束，并在新的一行开始输入下一个数组的数据。

例14 由键盘输入向量A：$(1,\ 3,\ 5,\ 7,\ 9,\ 11,\ 13,\ 15)'$ 和矩阵B：$\begin{pmatrix}2 & 4 & 6\\ 8 & 10 & 12\end{pmatrix}$。

```
!
DIM a(8),b(2,3)
MAT INPUT a,b
MAT PRINT a,b
END
......................................................................
OK.RUN
? 1,3,5,7,9,11
Too few input items. Please add more.
? 13,15
? 2,4,6,8,10,12,14
Too many input items. Excess ignored.
1           3           5           7           9
11          13          15
2           4           6
8           10          12
```

“？”和提示文字是自动显示的，就是说给的数据不够或太多时，系统都会作出反映。

4.MAT INPUT 语句调整数组上界

用DIM语句定义数组时，数组的维数和维界都只能用常数表示，即在DIM语句中已确定了数组的大小及维界。但有时也希望能对数组各维的上界进行调整，以改变数组的大小，适应处理的需要。这种调整可以用MAT INPUT语句实现。其实现方式有两种：

(1) 固定方式

例如，DIM A(5)

MAT INPUT A(3)

在MAT INPUT语句中可重新指定数组的新上界，这样数组A就被重新定义为含三个元素的数组。运行时只需且只能输入三个数据。

(2) 随机方式，在程序运行中，按实际需要，临时决定如何调整参数组上界。作法是：

DIM A(1)

MAT INPUT A(？)

运行时输入多少个数据，则数组A就接受多少个数据，即数组的上界是不固定的，由输入数

据个数确定，但这种方式只适用于一维数组。

在运行时临时指定数组上界的作法是：

```
DIM     A(1), B(1, 1)
INPUT   PROMPT "n, m=" :n, m
MAT     INPUT  A(n), B(n, m)
```

即在程序运行时，用临时键入的数据作为数组的上界，这样的处理方法灵活、实用。

在程序中将数组的维界设置成可调整的，可以节省内存空间，又可增强程序的灵活性与通用性。

5·MAT LINE INPUT语句

MAT LINE INPUT语句可以看成是MAT INPUT和LINE INPUT语句的组合，它把输入的每一行读入一个元素，直到读入足够的行为止，所以它使用的数组必须是字符串数组，语句的一般格式为：

MAT LINE INPUT 〈数组名表〉

MAT LINE INPUT prompt expr$: 〈数组表〉

其中数组调整只能用指定固定上界的方法进行调整，不能使用（?）形式调整数组。

例15

```
SET MODE "hires"
DIM a$(5)
INPUT prompt "输入数组元素数: " : n
MAT LINE INPUT a$(n)
PRINT
MAT PRINT a$
END
OK.RUN
输入数组元素数: 3
? 中华人民共和国
? 北京
? 兵器工业出版社
中华人民共和国    北京     兵器工业出版社
```

二、矩阵赋值与运算语句

1.矩阵赋值语句

矩阵赋值语句可以实现对数组的每个元素同时赋值。这对于数组的初始化是很方便的。

格式：MAT 〈数组名〉=〈表达式〉

MAT 〈数组名〉=〈数组名〉

MAT赋值语句允许用常量、变量、表达式或其它数组给一个数组赋值。

例如：DIM A(10), B$(5)

MAT A=0

MAT B$=" "

前一个语句将数组A的每个元素都赋以0，后一个语句将数组 B$ 的每个元素都赋为空串。

当应用MAT赋值语句，用数组对数组赋值时，要求它们的维数相同。即以一维数组对一维数组赋值，二维数组对二维数组赋值。如果赋值与被赋值的数组大小不相同时，系统会自动调整被赋值数组的各维上界，使之与赋值数组元素个数相同。

16例

```
! Matrix assignments.
DIM a(5 to 6, 5 to 6), b(30, 20)
MAT b=3
MAT a=b
PRINT, “Lower bound to” , “Upper bound” , “Size”
PRINT “a” , Lbound(a, 1), Ubound(a, 1)
PRINT, Lbound(a, 2), Ubound(a, 2), size(a)
PRINT
PRINT, “Lower bound to” , “Upper bound” , “Size”
PRINT “b” , Lbound(b, 1), Ubound(b, 1)
PRINT, Lbound(b, 2), Ubound(b, 2), Size(b)
END
OK.RUN
          Lower bound to     Upper bound     Size
a            5                  34
             5                  24            600
b         Lower bound to     Upper bound     Size
             1                  30
             1                  20            600
OK.
```

可以看出，执行MAT a=b语句后，数组a已被调整，数组的第一维的上界变为34，使其与数组b有相同的元素个数；同样 a 的第二维的上界变为24，而使第二维中共有20个元素。a的上界被扩展，共有600个元素，但a的下界没有被改变。

2.矩阵运算

(1) 矩阵加、减运算

在True BASIC中，可以用MAT语句作矩阵的加、减运算。运算遵从数学上矩阵运算法则：要求参加运算的矩阵行、列数相同；位于相同行、列上的对应元素作加、减。例，A，B，C表示三个具有相同维数及大小的数组，则可用下述语句

```
MAT  A=B+C
MAT  A=B-C
```

表示矩阵的加减运算，下述语句也是正确的。

```
MAT  A=A+B
```

如果被赋值数组维数不满足矩阵运算后的维界要求，则自动被调整。

例17 求数组A(5)与B(5)的和与差。

```
! Array Sums and differences
DIM a(5), b(5), sum(5), dif(5)
MAT READ a, b
MAT sum=a+b
MAT drf=a-b
MAT PRINT sum, dif
DATA 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
END
OK.RUN
7          9          11          13          15
-5         -5         -5          -5          -5
```

（2）矩阵与一个常数或变量相乘

若A、B是数组，n是常数或变量名，MAT A=n*B 表示用n乘以数组B的每个元素后，再对数组A的相应元素赋值。显然，数组A与B应具有相同的维数。

（3）矩阵乘

当A、B、C是矩阵时，

MAT A=B*C

表示将矩阵B与矩阵C的乘积赋于矩阵A。矩阵相乘遵从数学上的法则。即

若 B矩阵是L行M列矩阵

C矩阵是M行N列矩阵

则 A矩阵是L行N列矩阵

$$矩阵A的元素Aij=\sum_{k=1}^{m} Bik\cdot Ckj$$

例18 数组b(2、3) 与数组c(3，4) 的乘积是2×4的数组即A(2，4) 。

如：

1	2	3
4	5	6

b(2，3)

*

1	2	3	4
5	6	7	8
9	10	11	12

c(3，4)

=

38	44	50	56
83	98	113	128

a(2，4)

程序及运行结果为：

```
! Matrix multiplications
DIM b(2, 3), c(3, 4)
MAT READ b, c
MAT a=b*c
MAT PRINT a;
DATA 1, 2, 3, 4, 5, 6
DATA 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
END
OK.RUN
   38   44   50    56
   83   98   113   128
```

三、矩阵常数与函数

1.矩阵常数

在True BASIC中存放了一些数组常数，这些常数可以用来向数组赋值，并且还可以重新调整被赋值数组的大小。

(1) 常数zer

zer表示元素均为0的矩阵。

```
若有    DIM   A(5,5)
        MAT   A=zer
```

执行这个语句时，A被赋值为元素均为0的数组，维数和上、下界不变。

```
若    MAT   A=zer(10, 8)
```

赋值后，A成为10×8的二维数组，元素均赋值为0。

(2) 常数con

con表示元素均为1的矩阵

```
若有    DIM   B(12)
        MAT   B=con
```

赋值后，数组B的元素均为1。

用常数zer和con对数组赋初值是很方便的。

(3) 常数idn

idn是行，列数相同的方阵，且其左上角到右下角的对角线上各元素均为1，其余元素均为0，这样的矩阵叫做单位矩阵。

若是idn(n)则表示一个n×n的单位矩阵

(4) 常数nul$

nul$是所有元素均为空串的字符型数组。

```
若有    DIM   C$(4), D$(3, 2)
        MAT   C$=nul$
        MAT   D$=nul$(5, 5)
```

赋值后，C$是含有4个空串元素的字符型数组，而D$成为含有5×5个空串元素的数组。

2.矩阵函数

(1) det (A) 函数值是矩阵A的行列式值，这时要求A是方阵。

(2) dot(A, B) 函数值是向量A与B的点积，即A与B是长度相同的一维矩阵，其点积是对应元素乘积之和，是一个数值。

(3) inv(A) 函数值是方阵A的逆矩阵。如果方阵A是奇异的，则逆矩阵不存在，系统将给出一个错误信息。

(4) trn(A) 函数值是矩阵A的转置矩阵。即若A是n×m矩阵，转置后得到一个m×n矩阵B，它们之间元素的关系是A(i, j)=B(j, i)。

四、矩阵的应用

在科技计算问题中，矩阵是十分有用的工具，在信中息管理工作中矩阵也得到广泛应用。在处理矩阵运算中，MAT类语句给我们提供了程序设计的有效手段。在此，我们介绍几个矩阵应用的例子，以启发读者的思路。

例19 假定一个班级最多有50个学生，每个学生最多有10个分数，计算每个学生的“加权平均分”。

要求解此问题，让我们先设几个数组，分数存放在数组“grades”中，权重放在“weights”中，算出的平均分放在“average”中，人名放在“name$”中。

所有的信息都放在DATA语句中。先读具体的学生数和每个学生的分数个数，然后用一个简单的MAT语句就可读入学生的全部分数。如：

```
MAT READ grades(s, g)
```

将读入s行g列的数组。

下面再用一个MAT语句就可算出所有平均值，最后打印出人名和平均分的表。

当学生人数很多时，通常把数据放在一个文件中，也可用MAT READ语句读入文件中的数据。

程序和运行结果为：

```
! ----Averaging grades
! Allow up to 50 students and 10 grades per student
dim grades(50, 10), weights(10), average(50), name$(50)
READ s, g
MAT read grades (s, g), weights(g), name$(s)
MAT average=grades*weights
FOR i=1 TO s
PRINT name$ (i) , average(i)
NEXT i
DATA 6, 4
DATA 87, 78, 82, 86
DATA 87, 99, 82, 100
DATA 73, 82, 67, 85
DATA 59, 62, 73, 81
DATA 88, 88, 88, 88
DATA 77, 61, 59, 93
DATA 0.2, 0.2, 0.2, 0.4
DATA Wang, Li, Zhang, Fun, Ling, Qing
END
OK.RUN
Wang                83.8
Li                  93.6
Zhang               78.4
Fun                 71.2
Ling                88.
Qing                76.6
OK.
```

例20 设已知三个商店均经营某四种商品，已知每种商品的单价，及各店每种商品的销售量。求各商店的营业额（只考虑这四种商品），三个店每种商品的总销售量，以及每个店各种商品销售量占该店总销售量的百分比。

这里用数组A(3，4) 存放各商店商品销售数据，数组M(4) 存商品单价，数组B(3) 存

销售量

品类 销量 店	商品1	…	商品4
店1	A_{11}	…	A_{14}
·	·		
·	·		
·	·		
店3	A_{31}	…	A_{34}

单价表

商品1	· · ·	商品4
M_1		M_4

各店营业额，数组C(4) 存各种商品的总销售量，数组D(3, 4) 存各个店每种商品销售量占该商品总销售量的百分比。

程序为：

```
SET MODE "hires"
DIM a(3, 4), m(4), b(3), c(4), d(3, 4)
MAT INPUT a(3, 4), m(4)
MAT b=a*m
PRINT "每店营业额"
MAT c=zer
MAT PRINT b
FOR j=1 TO 4
    FOR i=1 TO 3
        LET c(j)=c(j)+a(i,j)
    NEXT i
NEXT j
PRINT "每种商品总销量"
MAT PRINT C
FOR i=1 TO 3
    FOR j=1 TO 4
        LET d(1, j)=a(i, j)/c(j)*100
    NEXT j
NEXT i
PRINT "每店商品销量占总销量百分比"
MAT PRINT d
END
```

例21 为了科学地安排、预测和分析经济活动，投入产出法是一种常用的方法，用来研究各部门（各产品）之间生产与消耗的数量依存关系。这种依存不仅有直接的（以直接消耗系数矩阵表现出来)，同时还有间接的依存关系，若忽略了间接依存关系，将会造成很大的失误。将直接和间接的消耗依存关系合在一起，以矩阵形式表现出来，叫做完全消耗系数矩阵。

完全消耗系数矩阵的计算公式是：

$$B=(I-A)^{-1}-I$$

其中，A是直接消耗系数矩阵，是 n×n 方阵，n是部门（产品）个数；I是n×n单位矩阵；$(I-A)^{-1}$表式距阵I与A之差的逆矩阵；B是完全消耗系数矩阵，也是n×n矩阵。

下面给出的计算完全消耗系数的程序中，直接消耗系数矩阵是作为已知数据输入的。一般来说，直接消耗系数也是要经过计算才能得到的，那么，只要去掉这个输入，加上计算矩阵A的一段程序即可。另外，为了使程序通用，这里采用MAT类语句可调整矩阵大小的功能，程序运行时指定部门个数即可。

因为程序中不允许写成矩阵运算的表达式，所以上述运算要拆成几步来写，程序为：

```
SET MODE "hires"
DIM a(1, 1), b(1, 1), c(1, 1)
INPUT PROMPT "部门数" : n
MAT INPUT a(n, n)
MAT PRINT a
LET u=det(a)
PRINT u
IF u< >0 THEN
   MAT c=idn(n)
   MAT c=c-a
   MAT b=inv(c)
   MAT c=idn(n)
   MAT b=b-c
   PRINT "完全消耗系数矩阵"
   MAT PRINT b
END IF
END
```

习 题 六

1. 写出下列两个程序的运行结果，并说明为什么不同。

```
DIM a(4)                FOR K=1 to 4
FOR K=1 to 4              READ a1
   READ a(K)            NEXT K
NEXT K                  FOR K=1 to 4
FOR K=1 to 4              PRINT a1;
   PRINT a(K);            NEXT K
NEXT K                  DATA 37,24,45, 57
DATA 37, 24, 45, 57     END
END
```

2. 根据如下的DATA语句，用一维数组编写一程序，让屏幕上显示出有几个1，几个2，几个3和几个4。

```
DATA  1, 3, 4, 2, 3, 4, 1, 3, 4, 2, 3, 1, 3。
```

3. 编写一个程序将原顺序为10，11，6，15，33，31，6的各数按其大小顺序显示在屏幕上。
4. 编制一程序将五个人的姓名和编号存入计算机，今有人来查找他们，请计算机代为查人。无论有无此人均应给予答复。
5. 编一程序将杂乱无章的26个英文字母按字母表顺序显示出来。
6. 共有四个学生进行了五次测验（成绩由读者自行给定）。编一程序计算并显示各学生的总成绩及各自的平均成绩。
7. 写一程序，把A数组改变为B数组

$$A\begin{bmatrix}1 & 2 & 3 & 4\\5 & 6 & 7 & 8\end{bmatrix} \qquad B\begin{pmatrix}1 & 5\\2 & 6\\3 & 7\\4 & 8\end{pmatrix}$$

8. 求出一个任意二维数组中每行中元素值最大的元素，指出行号和列号。

9. 求出任意一个二维数组中某个元素，此元素在所在行上最大，在所在列上最小(称为“鞍”点)。这样的元素可能不止一个，也可能不存在。对于各种情况均应输出有关信息。

10. 某运输公司一周内各类车辆运行的公里数如表6-5所示。各类的车辆运费如表 6-6 所示。求出各类车辆一周的运费总额，全部车辆的运费总额，并用PRINT TAB语句打印一个表格。

11. MAT INPUT以何种次序给一个数组的元素赋值？试举例说明。

12. 下述两个语句有何区别？

MAT PRINT Array, MAT PRINT Array

13. 若数据定义为DIM A(3,3)，下述语句那些是正确的？

(a) MAT INPUT A(2, 2)　　(b) MAT INPUT A(5, 5)

(c) MAT INPUT A(2, 5)　　(d) MAT INPUT A(7:3, 2:5)

(e) MAT INPUT A(1:2, 1 to 3)　　(f) MAT INPUT A(7, 0 to 7)

表 6-5

星期 公里数 车类	一	二	三	四	五	六	七
大型卡车	1600	2300	4000	3600	7500	2400	3200
小型卡车	7500	6300	7200	4600	5900	6000	3500
小汽车	3000	2900	3100	2800	2700	3500	4000

表 6-6

运费 车型	元/公里
大型卡车	1.5
小型卡车	1.0
小汽车	0.8

14. 如果数组定义为DIM name$(100)，下述语句 MAT LINE INPUT Prompt"name$": name$ 是否在每一个name$输入前都显示提示？

15. 设A、B均为3×3矩阵，且已知它们的值：

$$A=\begin{pmatrix}1 & 3 & 5\\7 & 9 & 11\\13 & 15 & 17\end{pmatrix} \qquad B=\begin{pmatrix}2 & 4 & 6\\8 & 10 & 12\\14 & 16 & 18\end{pmatrix}$$

求：F=5*(A+B)*(A−B)。这里F也是3×3矩阵，试用矩阵语句和不用矩阵语句两种方法编写程序。

16. 设矩阵$A=\begin{pmatrix}5 & 3 & 1\\3 & 7 & 4\\1 & 4 & 9\end{pmatrix}$ 计算矩阵算式

$$P=I+A+A^2+A^3+A^4$$

这里I表示单位矩阵，$A^2=A*A$　$A^3=A*A*A$等等。问P的行、列数是多少？

17.利用逆矩阵解如下方程组。

$$\begin{aligned}
x_1-2x_2+3x_3-4x_4&=4\\
x_2-x_3+x_4&=-3\\
x_1+3x_2+x_4&=1\\
-7x_2+3x_3+x_4&=-3
\end{aligned}$$

第七章　函数和子程序

函数和子程序是True BASIC程序设计的辅助形式之一。

调用函数可以得到一个值、有一种是True BASIC系统内部定义的函数，叫做标准函数，不需说明就可以直接调用；另一种则是必须按要求用True BASIC语句进行定义后才可调用的函数称为自定义函数。子程序是根据需要而编辑的特定格式的程序模块，它可以供程序在任何位置上调用。

§7-1　标准函数

True BASIC中提供的常用的标准函数，只要正确掌握后，将可以大大简化程序设计，并给你带来许多方便。

一、标准函数的定义与分类

1.什么叫标准函数

在True BASIC中预定义一些函数名称，使用它可以传递一个或多个参数，经过一定的运算或处理而获得一个新的函数值（返回值）。这种特殊的变换关系称之为函数。简言之，函数就是一种特定的运算。在程序中要使用某个函数时，只要写出函数名并给定必要的参数，便能得到一个输出。标准函数是由语言本身提供的内部程序，故又叫内部函数。例如：

SQR (x) 被称为求平方根的函数，x是一个表达式，也称为函数的参数。SQR (9) 是将9这个参数值代入求平方根的函数，其返回值（函数值）为3。

2.标准函数的分类

在True BASIC中标准函数可分为以下几类：

(1) 算术函数：它可完成一定的算术运算或变换。

(2) 三角函数：它能完成三角函数的运算。

(3) 字符串函数：它可以对字符串变量进行运算操作，而获得对字符串的不同处理。

(4) 变换函数：它可将字符串转换成对应的ASCII码值或者由一个ASCII码值转换成对应的字符，以及数值与字符串之间的变换等。

(5) 逻辑函数：它可以用来检查是否按了键盘上的一个键，READ语句是否有数据可读，以及检查文件是否结束等。

(6) 日期和时间函数：它可以给出机器中当前的日期和时间。

(7) 控制函数：它可用于屏幕上打印（显示）及键盘输入位置的控制。

3.函数的调用

True BASIC提供的标准函数，可以用所给函数名及必要的参数来调用，调用时该函数可出现在程序语句同类表达式中，现以SQR (x) 为例。

例：

```
PRINT SQR (g)
LET  b=5+SQR (16)
LET  a=SQR (b*9)
```

在执行各语句时，遇到函数名后，则调用该函数进行运算，并以返回值提供给表达式中的函数取值。函数有参数时，必须用一圆括号括起来，有多个参数时，参数之间必须用逗号（“，”）分开。

二、算术函数

1.算术函数

算术函数是对数值型常数、变量或表达式进行特定的数学运算，其取值仍为数值的函数。为简明易查，现将True BASIC语言中提供的标准算术函数列入表7-1中。

2.标准算术函数的应用举例

例1 判断任一输入数是奇数或偶数，当输入为0时退出。

```
!
SET MODE "HIRES"
DO
    INPUT prompt "输入一个数（小于等于零退出）：" n
    IF MOD (N,2)=0 THEN
        PRINT "输入数为偶数! "
    ELSEIF n>0 THEN
        PRINT "输入数为奇数! "
END IF
LOOP UNTIL n<=0
EAD
```

```
OK.RUN
输入一个数（小于等于零退出）:12
输入数为偶数
输入一个数（小于等于零退出）:25
输入数为奇数!
输入一个数（小于等于零退出）-20
```

例2 随机数应用。设计一个程序产生1～100之间的随机数提供猜数，当猜数者输入的数大于或小于产生的随机数时，提示猜数者数太大或太小，并要求重猜。

```
!
CLEAR
RANDOMIZE
LET x=int (100*rnd)+1
LET n=0
DO
   INPUT prompt "Enter your guess (1-100) " :num
   LET n=n+1
   IF num=x THEN
     PRINT "Your guess is right! "
      EXIT DO
 ELSE
     IF num>x THEN
        PRINT "Your number is begger! "
    ELSE
      PRINT "Your number is smaller! "
      END IF
      PRINT "Try agien!"
    END IF
  LOOP
  PRLNT "You guess" ;n;  "times"
  PAUSE 2
  END
```

程序中使用了Randomize语句，这一语句称为随机数种子语句，执行此语句后，RND的取值不可预测，不用Randomize语句时，第一个RND函数值总是固定的。

表7-1 算术函数

函数名	功能说明	应用例	函数值
ABS(x)	求x的绝对值	ABS(−15) ABS(10)	15 10
INT(x)	求不大于x的最大整数	INT(3.56) INT(−3.56)	3 −4
ROUND(x)	求x的四舍五入的整数	ROUND(4.2) ROUND(−5.2) ROUND(−5.6)	4 −5 −6
ROUND(x,n)	求x的第n位小数近似值，若n≥0，则近似值到小数右边的第n位(四舍五入)；若n<0则取小数左边的第n位近似值。	ROUND(5.235,1) ROUND(5.235,0) ROUND(15.235,−1) ROUND(15.235,−2) ROUND(15.535,−1)	5.2 5.0 15 20 16
TRUNCATE(x,n)	截取x的第n位有效数字，n>0,，截去小数右边第n位以后数字；若n<0，则截去小数左边第n位以后数字、并以零代之。	TRUNCATE(115.33,1) TRUNCATE(115.33,0) TRUNCATE(115.33,−1)	115.3 115 110
SQR(x)	求x的平方根、x必须大于等于零	SQR(9) SQR(1024)	3 32
MOD(x,y)	返回x被y除的余数。取值在 与y之间，y不可为0。	MOD(10,2) MOD(9,5) MOD(12,9)	0 4 3

（续）

函数名	功能说明	应用例	函数值
MAX(x,y)	求x,y中的最大数	MAX(5,9) MAX(5,−9)	9 5
MIN(x,y)	求x,y中的最小数	MIN(5,9) MIN(5,−9)	5 −9
LOG(x) LOG2(x) LOG10(x)	求x以e为底的对数， 求x以2为底的对数， 求x以10为底的对数， x取值应大于0	LOG(10) LOG2(10) LOG10(10)	2.30259 3.32193 1
EXP(x)	给 e^x 为LOG(x)的反函数	EXP(0) EXP(1)	1 2.71828
SGN(x)	求x的符号 x>0，取值为1 x=0，取值为0 x<0，取值为−1	SGN(100) SGN(0) SEN(−10)	1 0 −1
REMAINDER(x,y)	将x被y除取余数 y≠0,x,y可为小数 当x，y均大于零时，与MOD函数相同，其他情况结果符号与x一致，不一定在0与x之间	REMAINDER(5,2) REMAINDER(1.3,1) REMAINDER(−1,3) REMAINDER(7,−3)	1 .3 −1 1
RND	返回一个大于0小于1的伪随机数（等概分布）	RND RND RND	.55337 .022 .83856
PI	π的常数值	PI	3.14159…
EPS(x)	可影响x取值的最小数，即影响x精度的最小数。	EPS(0) EPS(1e13)	1.11254e-308 0.3115
DIVIDE(x,y,q,r)	x被y除，求商q及余量r。	它不是函数，须使用如下语句调用： CALL DIVIDE(10,3,q,r)	q=3 r=1

例3 用蒙特卡罗法求圆周率π值。

蒙特卡罗法（Monte-Carlo Method）是一种模拟随机事件发生的技术，它几乎可用于任何部门。蒙特卡罗法往往涉及大量计算。这里以求π的近似值为例，说明其用法。

已知圆面积公式$S=\pi R^2$，当$R=1$时$S=\pi$，见图7-1所示，落入$x^2+y^2=1$圆内（面积的1/4）的点数与落入正方形（面积为1）的点数之比（等于圆面积与正方形面积之比）即π值。

```
!
SET MODE “hires”
RANDOMIZE
 INFUT FROMPT “输入计算点数:” :n
FOR i=1 TO n
  LET x=rnd
  LET y=rnd
  IF x*X+y*y<=1 THEN LET k=k+1
NEXT i
PRINT “pi=” :4*k/n
END
OK.RUN
输入计算点数:1000
pi=3.1404
OK.
```

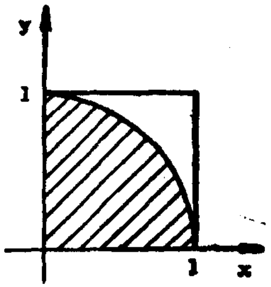

图 7-1

例4：生日问题。有人计算过50人一组的人中，两人在同月同日生的概率大于97%。这是一个有趣的概率问题。

算法分析：要计算一组N人中两人在同月同日生的概率P.可转化为先求该组中两人不同月同日生的概率q，则p=1−q

两人不在同月同日生的概率，等于该组人具有不同生日的总数与该组所有可能生日的组合总数之比，设后者为c，前者为d。

则$c=365\cdot365\cdots\cdots365=365^n$

$d=365\cdot364\cdots\cdots(365-n+1)$

则$q=d/c$

$p=1-q$

计算程序若设计为：

```
INPUT PROMPT “Enter the Size of group” :num
LET c=365^num
LET d=1
FOR i=365 TO num STEP-1
   LET d=d*i
NEXT i
PRINT “p=” ; 1-d/c
END
.........................................................................
Ok.PUN
Enter the size of group 50
Cverflow.
```

由于365^10数太大，使计算机产生溢出，故必须改用对数方法计算。可将上述程序改为：

```
!
INPUT PROMPT “Enter the size of group” :num
LET c_log=num*log(365)
LET d_log=0
```

```
FOR i=0 TO num-1
   LET d_log=d_log+log(365-i)
NEXT i
LET p*l-exp(d_log-c log)
PRINT "p=" ; p
END
.........................................
OK.RUN
Enter the size of group 50
p=.970374
OK.RUN
Enter the size of group 60
p=.994123
```

表7-1中最后一个DIVIDE(x，y，q，r)（求商及余数）不是函数，而是True BASIC提供的内部子程序。因为函数只能返回一个值，使用这一内部子程序的方法与子程序调用方法相同。下面给出一个具体的应用例子。

例5 将输入的十进制数变为二进制显示。

```
!
INPUT prompt "Enter the decimal number:" :a
LET s$= " "
LET num=a
DO
   CALL divide(a,2,q,r)
   PRINT q,r
   IF r=0 THEN LET w$= "0" ELSE LET w$= "1"
   LET s$=s$&w$
   LET a=q
LOOP UNTIL q=0
FOR i=len(s$)TO 1 STEP -1
   LET b$=b$&s$[i:i]
NEXT i
PRINT "Decimal number" ; num;  "translate to binary numer" ; b$
END
```

三、三角函数

True BASIC中提供的三角函数主要有sin(x)、cos(x)、tan(x)、atn(x)。在函数中x参数可允许使用不同的角度单位（度或弧度），若不加说明，所有角度均以弧度为单位。若你要使用度为单位（0～360°）时，要用option angle degrees语句预先声明，若再要以弧度表示时，则要采用option angle radians语句说明，其后的角度就均以弧度为单位（0～2π）了。

例6

```
!
CLEAR
```

```
OPTION ANGLE degrees
PRINT "tangent of 45 degrees is" ; tan(45)
OPTION ANGLE radians
PRINT "tangent of pi/4 radians is" ; tan(pi/4)
END
.....................................................................
OK.RUN
tangent of 45 degreea is 1
tangent of pi/4 radians is 1
```

程序中pi表示圆周率π的值，可由系统自动提供其准确值3.141592653……。为了变换方便，True BASIC中还提供了度与弧度变换函数DEG(x) 与 RAD(x) 。DEG(x) 是把以弧度表示的x值转变为角度值。RAD(x) 是把以角度表示的x值转变为弧度值。

为便于查询，现将True BASIC提供的三角函数列于表7-2中。

表7-2　三角函数

函数名	功能说明	调用例	函数值
sin(x)	求x角的正弦，x可用角度或弧度表示，用option Angle degrees (Radians)选择	sin(45) sin(PI/4)	0.707 0.707
cos(x)	求x角的余弦	cos(45) cos(PI/4)	0.707 0.707
tan(x)	求x角的正切	tan(45) tan(PI/4)	1 1
atn(x)	求x的反正切	atn(1)	45
deg(x)	将x的弧度值变为角度值	deg(PI/4)	45
rad(x)	将x的角度值变为弧度值	rad(1)	0.017453

四、字符串函数

True BASIC提供了字符串处理函数，这些函数可用于求字符串长度，求字符串在字符串中的位置，及将字母由大写变为小写（或相反）等等。在函数中A$、B$均为字符串的表达式，n为数值或数值变量。True BASIC提供的字符串函数列于表7-3中。

字符串函数应用，请参阅第九章。

五、变换函数

在程序设计中，经常需要ACSII码与对应字符之间的变换，字符串与数字之间的变换。现将True BASIC提供的这些变换函数列于表7-4中。

变换函数的应用，请参阅第九章。

表7-3 字符串函数

函数名	功能说明	调用例	函数值
LEN(A$)	求字符串A$中的字符个数，包括空格字符。	LEN(“Good morning”) LEN(“book”)	12 4
POS(A$,B$)	求B$字符串在A$字符串中的位置。	LEN A$=“ABCD” LEN B$=“B”	2
POS(A$、B$,n)	B$若出现于A$中，求B$的第一个字符在A$中的位置。n为可选参数，给定n时，函数由A$中第n个字符开始查找B$。无n时由第一个字符查找。如找不到B$时，则返回0。	POS(A$,B$) POS(A$,B$,3) POS(A$,“ ”,1)	2 4 0
UCASE$(A$)	将A$字符串中所有小写字母变为大写字母。	UCASE$(“good”)	“GOOD”
LCASE$(A$)	将A$字符串中所有大写字母变为小写，若字符串中有其它字符时，保持不变。	LCASE(“ABCD”)	“abcd”
REPEAT$(A$,N)	将A$字串重复N次组成新的长串，n>0，当n=0时产生空串，当n<0时，给出错误信息。	Repeat$(“*”,5) Repeat$(“*”,3)	“*****” “***”
TRIM$(A$)	除去A$字符串的首尾部空格。	LET A$=“ How are you?” TRIM$(A$)	“How are yo?”
LTRIM$(A$)	除去A$字符串左端的空格（首部）	Ltrim$(“ 123 ”)	“123 ”
RTRIM$(A$)	除去A$字符串右端（即尾部）的空格。	RTRIM$(“ 123 ”)	“ 123”
A$[m:n]	求A$字符串中由第m个字符开始，到第n个字符组成的子字符串。	LET A$=“How are you?” LET B$=A$[5:7] LET C$=A$[9:11]	“are” “you”
A$[M:MAXNUM]	求A$字符串由第m个字符到最后一个字符组成的子字符串，其中MAXNUM=LEN(A$)	LET D$=A$ D$[5:MAXNUM]	“are you?”

表7-4 变换函数

函 数 名	功能说明	调用例	函 数 值
CHR$(x)	将x数字表达式取值后变换成相应的ASCII码字符，（x值应在0～255之间，超过255对256取模）。	CHR$(65) CHR$(90) CHR$(66+255)	“A” “Z” “A”
ORD(A$)	将A$所代表的一个ASCII码字符，变换成为对应的ASCII码序值，A$长度超过1时出错。	ORD(“A”) ORD(“Z”) ORD(“B”)	65 90 66
VAL(A$)	将A$（为数字字符串，可包括+、-号及小数点）变为对应的数字值。	VAL(“-123”) VAL(“12.6”)	-123 12.6
STR$(x)	将数字表达式x的值变为相应的数字字符串，若x为指数形式，也变为非指数形式的字符串。	STR$(125) STR$(.5e-3)	“125” “.0005”

要换出数的应用，请参阅第九章。

六、日期和时间函数

如果你使用的机器上有一个时钟，你可以用函数得到当前的日期和时间。如果你的机器上没有时钟，则给出相应的标志，见表7-5所示。

表7-5 日期和时间函数

函 数 名	功 能 说 明	应 用 例
DATE	取当前日期，用yyDDD编码	88031即1988年第31 天
TIME	取当前时间，以午夜后的秒表示	342即00:54:02A.M
DATE$	当前日期yyyymmdd	yyyy——年 mm——月 dd——日
TIME	当前时间hh:mm:ss	hh——时 mm——分 ss——秒

1. 日期函数 date$

该函数可给出当前日期，其日期形式为YYYY MM DD，其中YYYY表示年，MM表示

月，DD数字表示日。

例如　PRINT DATA $

显示：1989 10 20这表明当前日期是1989年10月20日。

2.天数总计函数 DATE

该函数可计算出从当年元旦开始至今的总天数.其形式是“YYDDD”，YY表示年份，DDD表示天数。

例如　PRINT DATA

输出：89323　表示89年第323天。

3.时间函数 TIME$

该函数给出的时间形式为hh:mm:ss如果机器上无时钟，则time值为99:99:99。

例如　PRINT TIME$

输出：10:11:02这表明当前时间为10点11分02秒。

4.时间总计函数 TIME

该函数将给出由午夜0时至现在所经过的时间（以秒为单位），如果机器上无时钟，则time将为－1。

例如　PRINT TIME

输出：35461.3

§7-2　自定义函数

上节介绍了标准函数及其应用。在编程中使用标准函数可以节省很多编程时间，但仅有语言提供的标准函数往往是很不够的。True BASIC允许在程序中用定义的方式，把一个程序中需要多次为某个值进行的计算或操作过程定义成一个函数，这样，在同一个程序中，就可以用一个很简单的函数代替一长串复杂的表达式或一段程序。

一、函数定义格式

1.单行自定义函数

单行自定义函数。其格式为：

DEF〈函数名〉[（〈参数1〉，〈参数2〉，…）]=〈表达式〉

其中，DEF为自定义函数语句的关键字，表明以下为自定义函数行：〈函数名〉…是自定义函数命名，命名法与简单变量命名规则相同，字符串函数名最后一个字符必须是“$”；〈参数1〉，〈参数2〉，…是自定义函数中使用的参数，可以是数字变量、字符串变量或数组变量。

例7　用公式INT（10*RND）+1产生1—10之间的一个随机函数存入d：

DEF d=INT(10*RND)+1

在程序的表达式中再有d时，则按右部表达式计算其值。下面是调用五次d，并计算五次产生的随机数之和的源程序和运行结果为：

```
! 无参自定义函数
DEF d=INT(10*RND)+1
LET s=0
 FORi=1 TO 5
```

```
  LET abc=d
  PRINT :abc,
  LET s=s+abc
NEXT i
PRINT
PRINT "s=" ; s
END
OK.RUN
6    1    7    4    10
s=28
```

例8 计算$s=\sum_{i=1}^{n} i^2$直到s≥500时为止，并输出s值及项数n。

程序及运算结果为：

```
! 有参自定义函数
! s=1^2+2^2+3^2+…+n^2=n(n+1)(2n+1)/6
DEF bs(n)=n*(n+1)*(2*n+1)/6          ! 函数定义语句
LET k=1
DO
  LET s=bs(k)                        ! 调用函数
  IF s>=500 THEN EXIT
  LET k=k+1
LOOP
PRINT"s="; s,"n="; k
END
OK.RUN
s=506       n=11
```

2.多行自定义函数

在一行内不能实现一个自定义函数的定义时，须采用多行自定义函数，定义格式有两种：

(1) 无参数函数：

```
DEF〈函数名〉
       ⋮
LET〈函数名〉=〈表达式〉
       ⋮
END DEF
```

例如：计算百位随机整数与个位随机整数之差

```
DEF m
  LET a=INT(100*RND+100)
  LET b=INT(10*RND)
  LET m=a-b
END DEF
```

(2) 有参数函数：

```
DEF〈函数名〉(〈形参1〉,〈形参2〉,…)
                  ⋮
LET〈函数名〉=〈表达式〉
                  ⋮
END DEF
```

多行自定义函数由DEF开始到END DEF结束，组成一个自定义函数结构模块，两者缺一不可；在函数中的语句（…）可以是True BASIC语言的任何语句，根据函数定义需要选择。多行自定义函数体中至少有一个语句是给函数名赋值的语句，调用时函数最后的值由此赋值语句决定。

例9 计算 $s=\sum_{i=1}^{n} i^2$ 直到s≥500时为止，并输出s值及项数n。程序中，用程序段定义

$$\sum_{i=1}^{n} i^2$$

```
! 有参自定义函数块结构
! s=1^2+2^2+3^2+…n^2
DEF bs(n)                                  ! 函数定义语句
    LET b=0
    FOR i=1 TO n
       LET b=b+i2
    NEXT i
    LET bs=b
END DEF
LET k=1
DO
    IF bs(k)>=500 THEN EXIT DO             ! 调用函数
    LET k=k+1
LOOP
PRINT "s="; bs(k),"n="; k
END
Ok.run
s=506        n=11
```

程序中的bs（k）等价于执行如下一段程序：

```
LET b=0
FOR i=1 TO k
  LET b=b+i^2
NEXT i
LET bs=b
```

例10 定义一个字符串处理函数，可将由键盘输入的一个字符串的字符顺序按反向排列组成一个新字符串，并显示出来，如果输入字符串是“END”或“end”时，停止执行程序。

```
! ask fof string and print them in revers
!
! The reverse$ funcition
DEF reverse $(s$)
    IF s$="end"OR s$="END" THEN STOP
    LET result$=" "
    FOR i=Len(s$) TO 1 STEP -1
        LET result$=result$ & s$[i:i]
    NEXT i
    LET reverse$=result$
END DEF
DO
    INPUT PROMPT "Another string? " :a$
    PRINT reverse$(a$)
LOOP
END
```

二、函数定义规则

1.单行函数必须在同一个DEF语句中书写完，模块形式的函数，在DEF和 END DEF之间的程序模块中，最少须有一个给函数名赋值的语句。

2.函数名既是调用函数的标识符，又是存储函数值的变量单元。必须遵守变量名的命名规则，有数字型和字符型（$结尾）之分。

3.函数名后的圆括号及其括号中用逗号分隔的变量，叫函数的参数部分，其中的变量叫函数的参数。参数可以有一个或多个。不含参数部分的函数，叫作无参函数。

4.函数定义语句中的参数用于描述函数关系，叫形式参数，又叫形参、虚拟自变量或变元。只能用变量，不能用表达式或常量。变元只取形式，不分配内存单元，不接受数据，与程序中的变量无关。变元多于一个时，用逗号分隔。

三、函数调用规则

函数一经定义，就可以在同一个程序中多次被调用。函数调用应遵循以下规则：

1.和标准函数一样，只能在表达式中调用。在调用函数时，实在参数将取代形式参数，而且，实在参数的个数、类型必须与形式参数一一对应。

2.在调用函数时，计算机是这样执行的：先将实在参数的值传送到临时的内存单元中，再将临时变量的值进行运算。因此，若在调用函数过程中形式参数值发生变化，将不影响实在参数原来的值。

例11　输入任一正整数n，计算n!

为此，我们定义一个计算阶乘函数。

```
DEF mfactor (m)
    LET t=1
    DO
        LET t=t*m
        LET m=m-1
    LOOP UMTIL m=1
    LET mfactor=t
```

```
  END DEF
  INPUT PROMPT "n=" : n
  LET a=mfactor (n)
  PRINT "n! =" ; a
  END
```

若输入给n的值为8，在调用函数mfactor的过程中，m的值不断变化。调用结束时，m的值为1，由于n和m是两个不同的内存单元，因此，m值的变化并不影响n的值，调用结束后，n的值仍为8。

3．一般情况下函数必须先定义后调用，如果定义置于调用之后，则必须给函数命名：DECLARE DEF〈函数名〉，……

例12 把例7-9程序中的bs (n) 函数定义移到Loop语句之后，在调用之前有说明语句：DECLARE DEF bn，程序运行结束和例7-9程序运行结果相同。

程序及运行结果为：

```
! 有参自定义函数
! s=1^2+2^2+3^2+…n^2=n(n+1)(2n+1)/6
DECLARE DEF bs                              ! 函数命名语句
LET K=1
DO
  LET s=bs (k)                              ! 调用函数
  IF s>=500 THEN EXIT DO
  LET K=K+1
LOOP
DEF bs(n)=n*(n+1)*(2*n+1) /6                ! 函数定义语句
PRINT "s=" ; s, "n=" ; k
END
OK.RUN
s=506     n=11
```

4．函数名字不允许与程序中的变量同名。函数语句块中的非形参变量和主程序中的同名变量共享内存单元。形参不占内存单元，与主程序中的变量无关。

例13 计算$\sqrt{|x+y|}$

程序及运行结果为：

```
DEF abc(b,c)
    LET a=b+c
    LET abc=SQR(ABS(a))
END DEF
LET x=10
LET y=-200
PRINT "abc=" ; abc(x,y)
PRINT "a=" ; a
END
OK.RUN
abc=13.784
```

```
a=-190
```

在调用函数的过程中，由语句LET　a=b+c

在a变量中得到实在参数x+y的值，由程序中的输出语句打印出a=－190。

因为a不是函数的形参数，所以与程序中的a是同一个变量。

例14　计算$\sqrt{|x+y|}$，在函数中设参数a，在调用过程中存放b+c。

程序及运行结果为：

```
 DEF abc(a,b,c)
    LET a=b+c
    LET abc=SQR(ABS(a))
    PRINT "函数内部输出 a=" ; a
 END DEF
 LET x=10
 LET y=-200
 LET sa=abc(w,x,y)
 PRINT "主程序中输出  abc=" ; sa
 PRINT"                  a="; a
 PRINT"                  w="; w
 END
 Ok.RUN
 函数内部输出  a=-190
 主程序中输出  abc=13.784
               a=0
                w=0
```

内部函数abc (a, b, c)中,有一个形参赋值的语句：LEF a=b+c,主程序中调用函数语句：LEF sa=abc (w, x, y) 。在调用函数时，执行函数定义的程序块的PRINT："函数内部输出：a=" ；a输出a=－190，是由实参x－y得到的。当返回主程序时，输出a=0，因为a单元中没有数据，这说明形式参数在函数定义语句之外没有实际意义，仅限于函数内有效。输出w=0则说明函数的参数不能接收数据，也就是说，不能用参数向程序回带值。

5.在函数定义中，可以使用EXIT DEF语句，它可以出现在函数定义中的任何地方。执行此语句时，函数就返回到调用处，形成函数的非终端出口。

例15　用函数计算　$C_m^n=\frac{m!}{n!\ (m-n)!}$

```
DEF fact(n)
    IF n=0 OR n=1 THEN
       LET fact=1
       EXIT DEF
    ELSE
       LET f=1
       FOR i=1 TO n
           LET f=f*i
```

```
          NEXT i
          LET fact=f
          END IF
      END DEF
  PRINT "Enter two possitive integer: ";
  INPUT PROMPT "n,m=" : n,m
  LET c=fact(m)/(fact(n)*fact(m-n))
  PRINT "c="; c
  END
```

上述程序除了在函数定义中使用EXIT DEF语句，还说明了函数定义中参数n与程序中的变量n是两个不同的变量，它们是彼此独立的，无任何关系。

6.在定义函数时，若用数组作为形式参数，要用括号和逗号表示数组维数：

a ( )　　表示a是一维数组
a (,)　　表示a是二维数组
a (,,)　　表示a是三维数组

以此类推。

在调用函数时，括号内只写数组名，而不必用逗号表示维数。

例16　读入一维数组，输出下标变量的全部数据及其和。

程序及运行结果为：

```
  DEF aaa(a(  ),n)              ! 形参a是一维数组
      FOR i=1 TO n
      READ a(i)
      PRINT a(i)
      LET abc=abc+a(i)
      NEXT i
      LET aaa=abc
  END DEF
  DIM x(5)
  LET s=aaa(x,5)          ! 定义数组
  PRINT "    s="; s
  DATA 1,2,3,4,5
  END
  OK.RUN
  1
  2
  3
  4
  5
      s=15
```

用数组传送数据时必须谨慎，因为每次调用函数时，整个数组被拷贝到一个临时数组中，所以有点慢，并且比在子程序中使用要占更多存储单元。

7.允许函数嵌套调用，即一个函数中调用另一个函数，乃至多层函数嵌套调用，也可以

用EXIT DEF跳出函数到调用的下一个语句。

例17 已知三角形的边长，用边公式计算三角形的面积，

$$p=0.5(a+b+c)$$

$$s=\sqrt{p(p-a)(p-b)(p-c)}$$

当三边a，b，c的值不能组成三角形时，输出＊＊＊＊＊＊标志。

流程图见图7-2所示。

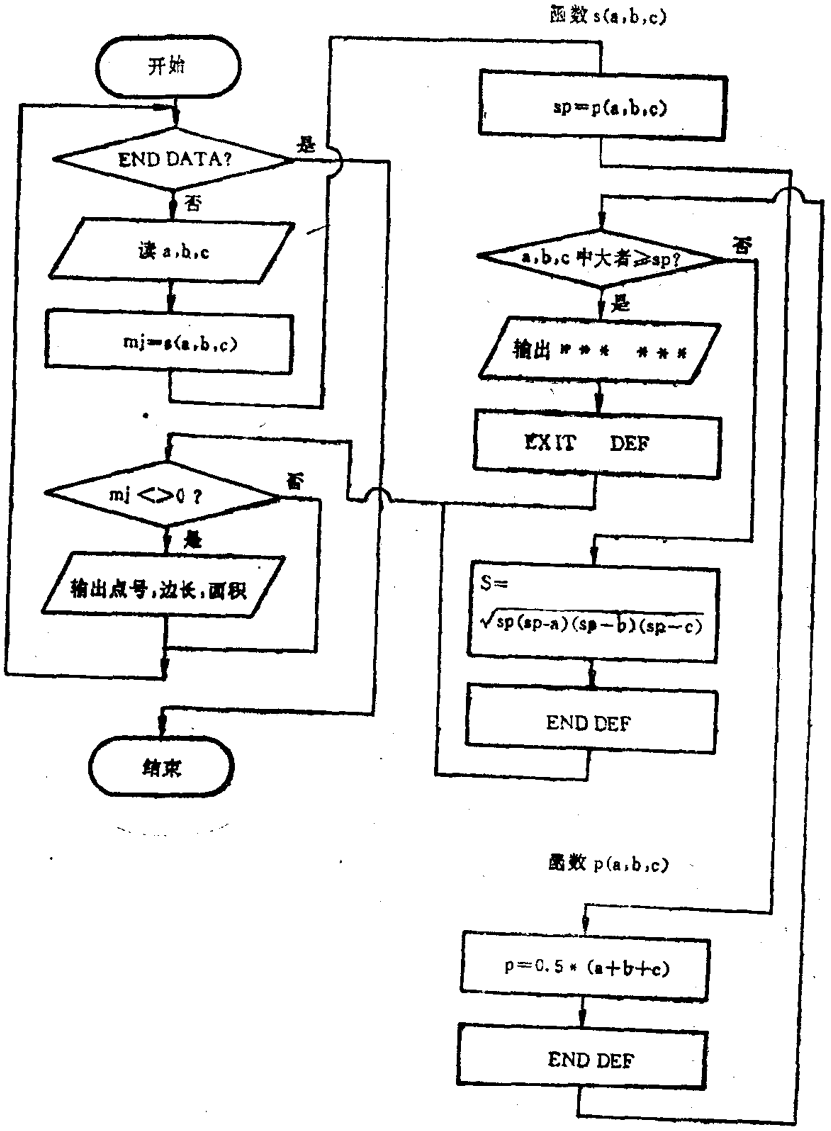
函数 s(a,b,c)
开始
END DATA?
是
否
读 a,b,c
mj=s(a,b,c)
mj <>0 ?
否
是
输出点号，边长，面积
结束
sp=p(a,b,c)
a,b,c 中大者≥sp?
否
是
输出＊＊＊ ＊＊＊
EXIT DEF
S=√sp(sp-a)(sp-b)(sp-c)
END DEF
函数 p(a,b,c)
p=0.5＊(a+b+c)
END DEF

图 7-2

程序和运行结果为：

```
! 三角形的面积=SQR(p*(p-a)*(p-b)*(p-c))
DEF p(a,b,c)=0.5*(a+b+c)
DEF s(a,b,c)
    LET sp=p(a,b,c)
    IF MAX(a,MAX(b,c))>=sp THEN
       PRINT "第"; n; "个，不成三角形******"
       PRINT
```

```
      EXIT DEF              ! 跳出函数体到调用的下一个语句
      END IF
      LEF s=SQR(sp*(sp-a)*(sp-b)*(sp-c))
   END DEF
   DO UNTIL END DATA
      LET mj=0
      READ n,a,b,c
      LET mj=s(a,b,c)
      IF mj<>0 THEN
         PRINT"第"; n; " 个图形边长: "; a,b,c
         PRINT "        面积: s="; mj
         PRINT
      END IF
   LOOP
   DATA 1,23,15,34,2,28,38,56,3,34,98,67,4,100,34,50,5,4,5,6
   END
   OK.RUN
      第1个图形边长: 23            15            34
            面积: s=140.2
      第2个图形边长: 28            38            56
            面积: s=481.139
      第3个图形边长: 34            98            67
            面积: s=563.663
      第4个, 不成三角形*******
      第5个图形边长: 4             5             6
      面积: s=9.92157
```

这个程序中，通过语句：

LET mj=s(a,b,c)

调用自定义函数s。在调用过程中，又通过语句：

LET sp=p(a,b,c)

调用了函数p，构成了函数嵌套调用。

这个程序中两个自定函义数的形参都采用了a，b，c，调用时实参也是a，b，c。这说明实参与形参可以同名。但必须注意，即使同名，形参和实参也是完全不同的含意，形参只是一种形式符号，而实参则必须是已被赋值的变量或数据。

另外，程序中的LET mj=0语句可以缺省，因为函数每次被调用时，函数名字的单元都经初始化为零，如此，调用函数s(a，b，c)时，如果由于三边数据不能组成三角形，使语句：s=SQR (sp*(sp-a)*(sp-b)*(sp-c))不被执行，s保持初始值0，而不是前一次被调用所得到的值。

8.自定义函数允许自归调用。即函数自己调用自己。

(1) 定义函数呈自归调用形式。

例18 把n! =n·(n-1)·(n-2)…2·1计算过程定义成函数：

DEF fact(n)

```
    IF n<=1 THEN fact=1 ELSE
fact=n*fact(n-1)
    END DEF
```

函数计算见流程图7-3所示。

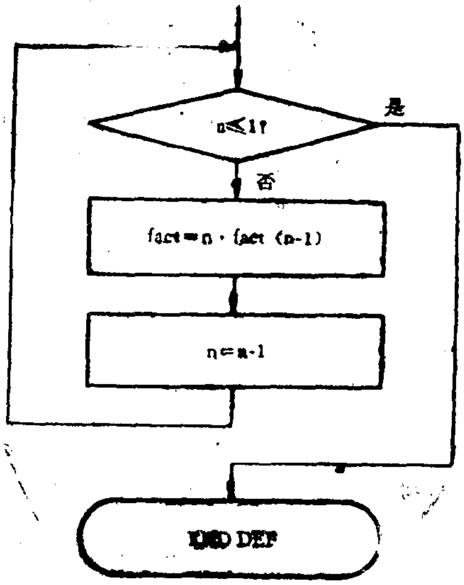

图 7-3

这就是一个自归调用的函数。

调用函数fact(n)，计算1，2，3，4，…，9，10等10个数的阶乘。

程序及运行结果为：

```
! 函数自归调用
DEF fact (n)
    IF n<=1 THEN LET fact=1
    ELSE LET fact=n*fact(n-1)      ! 自归调用
END DEF
FOR i=1 TO 10
    PRINT "i="; i, "fact (" ; i;  ") =" ; fact(i)
NEXT i
END
OK.RUN
i=1         fact(1)=1
i=2         fact(2)=2
i=3         fact(3)=6
i=4         fact(4)=24
i=5         fact(5)=120
i=6         fact(6)=720
i=7         fact(7)=5040
i=8         fact(8)=40320
i=9         fact(9)=362880
i=10        fact(10)=3628800
```

(2) 调用函数时的自归调用。调用函数时，自变量可以是函数自身。

例19 定义函数fact(x)$=x^2$，用自归调用方式计算3^{16}。

$3^{16}=3^{2\times2\times2\times2}$，嵌套自归调用4次即可得到。

源程序及运行结果：

```
! 函数自归调用
DEF fact(x)=x^2
LET s1=fact(fact(fact(fact(3))))
LET s2=(((3^2)^2)^2
PRINT "s1= "; s1," s2=" ; s2," 3^16= "; 3^16
END
OK.RUN
s1=43046721     s2=43046721     3-16=43046721
```

四、应用举例

例20 已知任意四个点的坐标，计算四边形abcd的面积，见图7-4所示。

作辅助线ac，用边公式分别计算△abc和△acd的面积，再计算两个三角形面积之和。用单行函数定义计算边长公式：

$$l(x1, y1, x2, y2)=\sqrt{(x1-x2)^2+(y1-y2)^2}$$

用多行函数定义三角形面积的计算公式：

$$P=0.5(a+d+c)$$

$$mare=\sqrt{p(p-a)(p-b)(p-c)}$$

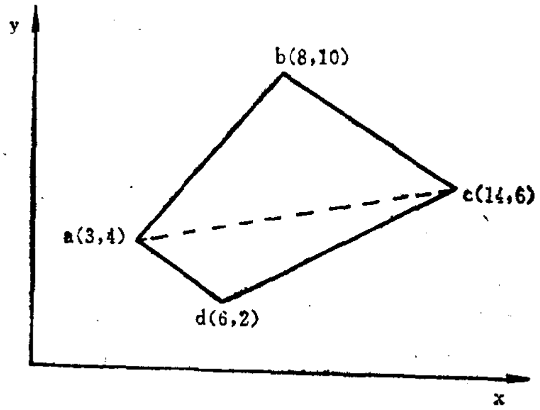

图 7-4

```
DEF mare(a,b,c)
    LET p=0.5*(a+b+c)
    LET mare=SQR(p*(p-a)*(p-b)*(p-c))
END DEF
DEF l(x1,y1,x2,y2)=SQR((x1-x2^2+(y1-y2^2)
READ xa,ya,xb,yb,xc,yc,xd,yd
    LET lad=l(xa,ya,xb,yb)
    LET lbc=l(xb,yb,xc,yc)
    LET lcd=l(xc,yc,xd,yd)
    LET lda=l(xd,yd,xa,ya)
    LET lac=l(xa,ya,xc,yc)
    LET s=mare(lab,lbc,lac)+mare(lac,lcd,lda)
PRINT" 四边形abcd的面积: " ; s
DATA 3,4,8,10,14,6,6,2
END
OK.RUN
四边形abcd的面积: 42
```

例21 形参是字符串，实参也必须是字符串。下面是输出字符串倒序排列的函数，函数值（倒序串）由函数名字reverse$带回到程序中。

程序和运行结果为：

```
! 串倒序
DEF reverse$(s$)
    LET result $=" "
    FOR i=LEN (s$) TO 1 STEP -1
        LET result $=result$ &s$[i:i]
    NEXT i
    LET reverse$=result$
END DEF
DO WHILE MORE DATA
    READ a$
    PRINT "字符串: " ; a$
    IF a$= " " THEN EXIT DO
    PRINT" 倒序串: " ; reverse$ (a$)
LOOP
PRINT "      ****   END   ****"
DATA book,Draw squarc and stop when user points the curdor into, " "
END
OK.RUN
    字符串: book
倒字串: koob
    字符串: Draw squarc and stop when user points the curdor into
倒符串: otni rodruc eht stniop resr nehw pots dna crauqs warD
    字符串:
            ****END****
```

例22 利用梯形法求定积分。

设函数f(x)在[a, b]区间上连续，且f(x)⩾0，求以曲线y=f(x)为曲边，底为[a, b]的曲边梯形面积A，A表示为定积分。

$$A=\int_a^b f(x)\,dx$$

利用梯形法求定积分的公式为：

$$\int_a^b f(x)\,dx \approx \sum_{i=0}^{n-1}[f(a+i\cdot h)+f(a+h\cdot(i+1))]\cdot h\div 2$$

$$=h\left[\frac{f(a)+f(b)}{2}+\sum_{i=1}^{n-1}f(a+i\cdot h)\right]$$

其中，h为步长，且h=(b−a)/n，n为分割区间[a,b]的等份数。

试利用梯形法求解，见图7-5所示的阴影区的面积。为了确定积分的上下限，求解联立方程组

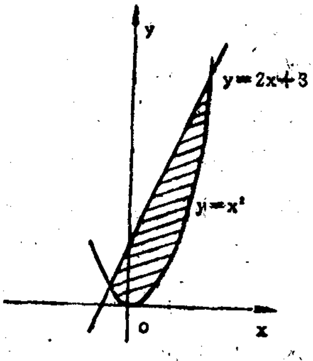

图 7-5

$$\begin{cases} y=2x+3 & (1)\\ y=x^2 & (2)\end{cases}$$

由此解出，$x_1=-1$，$x_2=3$，故积分区间为[−1, 3]，我们把该区间分成任意等分，即可确定步长h。

这里，在程序设计时，把被积函数作为一个自定义函数fan(x)。

程序及运行结果为：

```
 DEF fan(x)
LET f1=2*x+3
 LET f2=x^2
LET fan=f1-f2
END DEF
INPUT PROMPT"a,b,n? ":a,b,n
LET h=(b-a)/2
LET S=(fan(a)+fan(b))/2
FOR i=1 TO n-1
  LET S=S+fan(a+i*h)
NEXT i
LET S=h*S
PRINT"S="; S
END
OK.RUN
a,b,n? -1,3,40
S=10.66
```

§7-3 子 程 序

在§7-2介绍了自定义函数，程序中遇到反复使用而书写又比较复杂的算术或字符型表达式，用户可自己定义一个函数，使用简单方便，尤其模块形式的函数，还可以有辅助操作，调用过程中还可以得到其它的值，拓宽了函数的使用范围。但是它的定义和使用有一定局限性。①函数被调用时，函数名得到一个值，对于仅有操作而不产生值的问题不能使用；②函数的自变量只能是数据的源，不能接受和存放数据，函数值仅通过函数名字回带，实现函数程序模块，与主程序中的数据交换不够灵活。

True BASIC提供的另一类过程结构是子程序。子程序是一个相对独立的程序段，完成一定的功能，其他程序需要这个功能时，可以多次“调用”它。子程序完成自己的工作后，会自动地“返回”调用它的地点，使调用它的那个程序能够继续执行。调用子程序的程序，相对于子程序来说，称为主程序。

主程序调用子程序和从子程序返回主程序是怎样一个过程呢？

当主程序的某一语句处调用子程序时，计算机便停止执行主程序而转至子程序的入口处，开始执行子程序，在停止执行主程序的同时，把主程序停止处的下一条语句的位置保存了起来，这个位置称为断点。当子程序执行到它的返回出口时，便找出保存的断点，从而返回到主程序断点，使主程序继续执行，而断点的保存和恢复是由计算机自动完成的。

一、子程序定义规则

1.无参子程序

```
SUB<子程序名>
 语句块
END SUB
```

2.有参子程序

```
SUB<子程序名> (<形参1>, <形参2>, …)
语句块
END SUB
```

子程序定义必须以SUB语句开始，而且必须以END SUB语句结束。其中，子程序名是用户自己定义的子程序名，而形参1，形参2，…为子程序的形式参数，它们必须用括号括起来，它们之间以逗号分隔。SUB语句与END SUB语句之间的语句块为该子程序所完成的特定功能。子程序可以带参数也可以不带参数。

例23 调用子程序输出图7-6的图形。

程序及运行结果为：

```
! 无参子程序
SUB PP
   PRINT "*******"
END SUB
FOR i=1 TO 8
  PRINT TAB (i) ;
   CALL PP
NEXT i
END
OK.RUN
```

```
*******
 *******
  *******
   *******
    *******
     *******
      *******
       *******
```

图 7-6

例24 有参子程序pp (n)，输出一个＊。用不同的n进行不同方式调用，输出图7-7所示图形。

程序及运行结果为：

```
! 有参子程序
SUB pp (n)
```

```
    FOR a=1 TO n
      PRINT"*";
    NEXT a
    PRINT
  END SUB
  FOR i=1 TO 8
    PRINT TAB (i)
    CALL pp (8)
  NEXT i
  FOR i=1 TO 8
    PRINT TAB (10-i) ;
    CALL PP (i)
  NEXT i
  END
  OK.RUN
```

```
********
 ********
  ********
   ********
    ********
     ********
      ********
       ********
        *
       **
      ***
     ****
    *****
   ******
  *******
 ********
```

图 7-7

注意几点：

(1) 子程序名必须符合变量命名规则；

(2) 子程序名仅是调用的标识符，不能用作变量单元存储数据，没有类型区分，不允许尾部用$。

(3) 和有参函数相同，有参子程序的参数是虚拟变量，又叫形参，可以是字符型也可以是数字型。形参只能是变量或注明维数的数组。

(4) 形参可以从主程序向子程序传送数据，也可以由子程序向主程序传递数据。

二、子程序调用规则

1.必须用CALL命令调用。

无参子程序调用格式：

CALL〈子程序名〉

有参子程序调用格式：

CALL〈子程序名〉(〈实参1〉,〈实参2〉, …,)

2.调用无参子程序时，直接从SUB后的第一个语句开始执行，直到EXIT SUB 语句或END SUB语句返回主程序中本次调用语句的下一个语句，继续执行主程序。

例25 用子程序xy，交换x，y两个单元的数据，用子程序sub1输出 * * *。为了交换a和b中两个单元的数据，必须分两步实现：首先把a和b中的数分别送入x 和 y，调用子程序xy，再把x和y单元的数分别送入a和b，才实现了a和b单元数据的交换。

程序及运行结果为：

```
! 无参子程序及调用
! 交换x，y值的子程序
SUB xy
  LET w=x
  LET x=y
  LET y=w
END SUB
```

```
SUB sub1
    PRINT  "        * * *        ";
END SUB
PRINT TAB (8) ; "交换前"; TAB (42) ; "交换后"
PRINT   "........................................................"
FOR i=1 TO 5
    READ a,b
    PRINT a,b;
    CALL sub1              ! 调用子程序sub1
    LET x=a
    LET y=b
    CALL xy                 ! 调用子程序xy
    LET a=x
    LET b=y
    PRINT a,b
NEXT i
PRINT"........................END................................"
DATA 106,49,50,32,78,456,34,98,123,34
END
OK.RUN
```

```
交换前                          交换后
..............................................................................
106    49        * * *    49        106
50     32        * * *    32        50
78     456       * * *    456       78
34     98        * * *    98        34
123    34        * * *    34        123
...............................END.......................................
```

3.调用有参子程序时，实参的个数，类型必须与形参一一对应，从主程序向子程序传送数据的参数必须已被赋值，从子程序向主程序传送数据的参数必须是变量，在子程序中最少有一个给变量赋值的语句。

例26 用有参子程序计算：

$$s=\frac{8!}{4!+5!}$$

程序及运行结果为：

```
! 有参子程序及其调用
! s=a! /(b! +c! )
SUB subs(sj,n)
! n!
LET sj=1
FOR i=1 TO n
  LET sj=sj*i
NEXT i
```

```
END SUB
LET s=0
READ a,b,c
CALL subs(s2,b)
CALL subs(s3,c)
CALL subs(s1,a)
LET s=s1/(s2+s3)
PRINT “a! =”; s1,”b! =”; s2,”c! =”; s3
PRINT”a! /(b! +c! )=”; s
DATA 8,4,5
END
RUN
a! =40320        b! =24        c! 120
a! /(b! +c! )=280
```

在程序中：

子程序SUBS（sj, n），把n！存入参数Sj

调用语句CALL SUBS（s1, a）得到s1＝a！

调用语句CALL SUBS（s2, b）得到s2＝b！

调用语句CALL SUBS（s3, c）得到s3＝c！

可见，子程序的参数可以接收数据，这和函数用法相同；又可以作为存放数据的单元，把数据回带到主程序中，这又和函数的参数不同，函数只能用函数名回带数据。

形参和实参名称可以相同，也可以不同，但形参与实参意义不同。

4.调用子程序时，数组参数只写数组名，不写维数。

例27 已知两个数组：

$$a=\begin{Bmatrix}1&0&0&0\\0&2&0&0\\0&0&3&0\\0&0&0&4\end{Bmatrix}\qquad b=\begin{Bmatrix}9&0&0&0\\0&8&0&0\\0&0&7&0\\0&0&0&6\end{Bmatrix}$$

计算：

$$d(i)=a(i,\ i)^2-b(i,\ i)^2$$

$$x=a(1,\ 1)^2+a(2,\ 2)^2+a(3,\ 3)^2+a(4,\ 4)^2$$

$$y=b(1,1)^2+b(2,\ 2)^2+b(3,\ 3)^2+b(4,\ 4)^2$$

程序及运行结果为：

```
SUB sub(x(,),n,dia())                 ! 内部子程序
   FOR i=1 TO n
   LET dia(i)=x(i,i)*x(i,i)
NEXT i
END SUB
DIM a(4,4),b(4,4),d(4),aa(4),bb(4)
FOR r=1 TO 4
   FOR j=1 TO'4
```

```
READ a(i,j)
PRINT“”; a(i,j);
NEXT j
PRINT TAB(50);
FOR j=1 TO 4
    READ b(i,j)
    PRINT“ ”; b(i,j);
NEXT j
PRINT
NEXT i
CALL sub(a,4,aa)
CALL sub(b,4,bb)
FOR i=1 TO 4
  LET d(i)=aa(i)-bb(i)
  LET x=x+aa(i)
  LET y=y+bb(i)
NEXT i
PRINT“a数组对角线元素平方和: ”; x
PRINT“b数组对角线元素平方和: ”; y
PRINT“a, b数组对角线元素平方差: ”
FOR i=1 TO 4
  PRINT d(i),
NEXT i
PRINT
DATA 1,0,0,0,9,0,0,0,0,2,0,0,0,8,0,0,0,0,3,0,0,0,7,0,0,0,0,4,0,0,0,6
END
OK.RUN
  1  0  0  0                9  0  0  0
  0  2  0  0                0  8  0  0
  0  0  3  0                0  0  7  0
  0  0  0  4                0  0  0  6
a数组对角线元素平方和: 30
b数组对角线元素平方和: 230
a,b数组对角线元素平方差:
-80             -60             -40             -20
```

三、应用举例

例28 breakup(old$,new$,delim$) 是一个处理字符串的子程序，调用该子程序，进行如下处理：

在字符串a$中，把与d$中第一个符号相同的符号以前的全部字符提出存入n$中，提出的部分用一个“＊”补充，再把补充“＊”后的a$输出。例如：

a$＝“Live and let live”

d$＝“ ”

在Live后边是一个空格，于是把Live存入n$，a$成为：“＊and let live”第一个Live的位置被赋值一个＊。

程序流程图见图3-8所示。

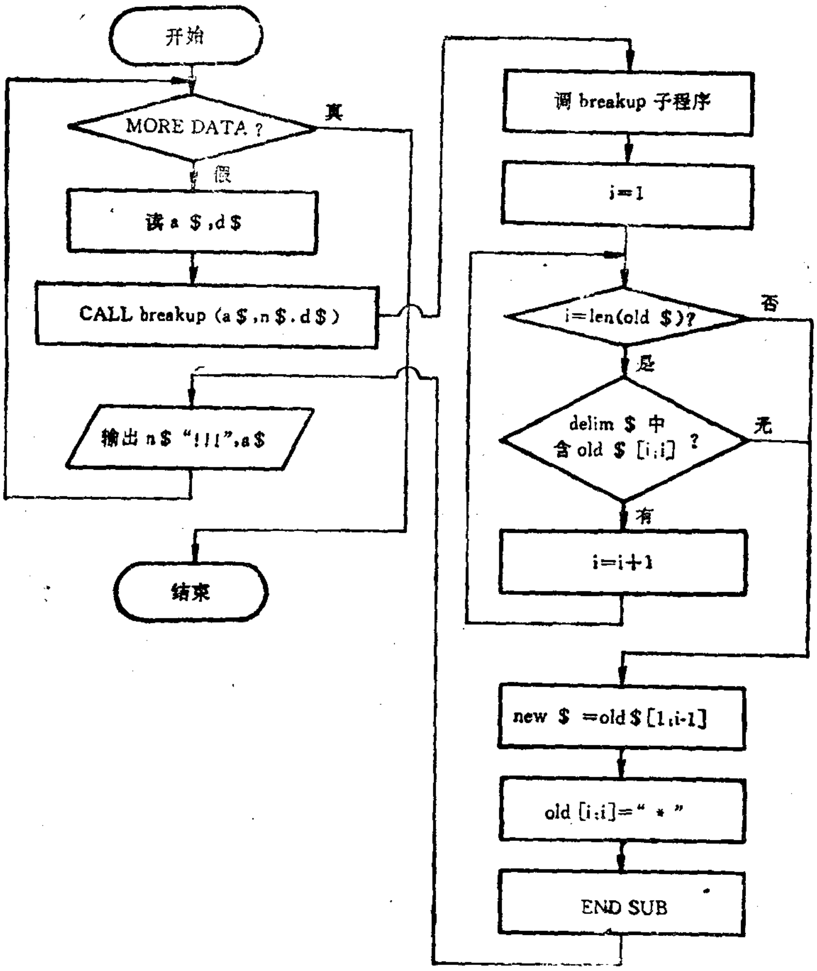
开始
MORE DATA ?
真
假
读a $,d $
CALL breakup (a $,n $.d $)
输出n $ "!!!",a $
结束
调 breakup 子程序
i=1
i=len(old $)?
否
是
delim $ 中
含 old $ [i;i] ?
无
有
i=i+1
new $ =old $ [1;i-1]
old [i;i]=" * "
END SUB

图 7-8

程序和运行结果为：

```
SUB breakup(old$,new $,delim$)
  FOR i=1 TO LEN(old$)
    IF POS(old$,delim$[i:i])<>0 THEN EXIT FOR
  NEXT i
  LET new$=old$)[1:i-1]
  LET old$[1:i]="*"
END SUB
DO WHILE MORE DATA
  READ a$,d$
  CALL breakup(a$,n$,d$)
  PRINT n$,"! ! ! "; a$
LOOP
DATA"Live and let live"," "
DATA"save,reset,catalog","; ,"
DATA"kept entire","; ,"
DATA"abcdefghi,and,book","an"
END
```

```
OK.RUN
Live             ! ! ! *and let live
save             ! ! ! *restet,catalog
kept entire      ! ! ! *
                 ! ! ! *bcdefghi,and,book
```

breakup是一个处理字符串的子程序，但其子程序名不允许有$ 标志，因为不能作存储数据的变量单元。

例29 计算 $1+\frac{1}{1\cdot2}+\frac{1}{2\cdot3}+\frac{1}{3\cdot4}+\cdots+\frac{1}{n(n+1)}$，直到最后一项的值小于某个小数。

设计一个完成1/n(n+1) 计算的子程序，结果存入s，精度值存入u，然后根据给定的精度要求，计算多项式的和并记下n值。

子程序的程序流程图见图7-9所示。

程序及运行结果为：

```
SUB abc(s,u)
    LET s=1
    LET n=1
    DO
        LET e=1/(n*(n+1))
        LET s=s+e
        IF e<u THEN EXIT SUB
        LET n=n+1
    LOOP
END SUB
DO
    READ a
    IF a=0 THEN EXIT DO
    CALL abc(s,a)
    PRINT"s="; s,"miu="; a,"n="; n-1
LOOP
DATA 1E-3,1E-5,1E-2,0
END
OK.RUN

    s=1.9697      miu=.001       n=31
    s=1.99685     miu=.00001     n=315
    s=1.90909     miu=.01        n=9
```

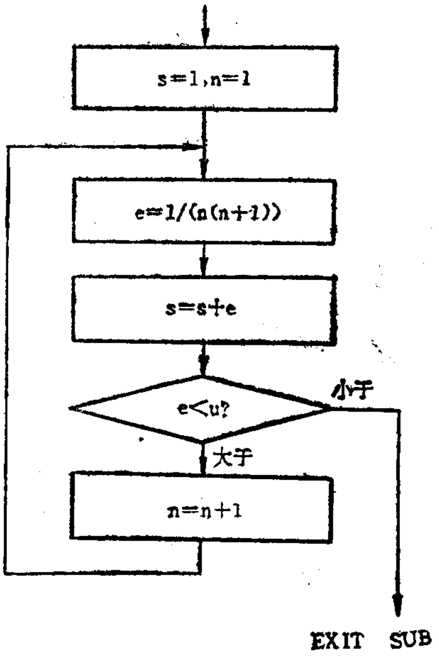

图 7-9

子程序中设置了IF e<u THEN EXIT SUB语句，用于中途退出子程序。

例30 用二分法求方程$x^3-1.5x-1=0$的根（精确到0.0001）。

用二分法求f(x)=0的根的原理为：

已知f(x)=0在区间[a，b]上有且只有一个实根，则f(a)与f(b)必异号，见图7-10所示。

我们令 $m=\frac{1}{2}(a+b)$ ；若f(m)与f(a) 异号，则根在a和m之间。否则，根在m和b之间。重复这个对分区间的过程，当区间长度小于等于给定精度e_0时，取该区间中点为根的近似值。

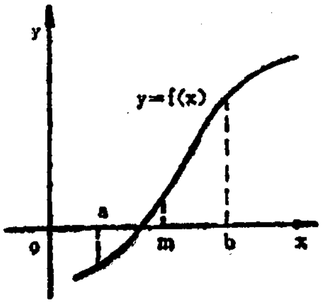

图 7-10

在程序设计时，我们定义一个子程序 binary(x1, x2, e)，功能是由区间的上、下界和精度求方程的根，还定义一个函数以决定f(x)。

程序及运行结果为：

```
! BINARY METHOD
DEF hanshu(x)=x^3-1.5*x-1
SUB binary(x1,x2,e)
    LET y1=hanshu(x1)
    LET y2=hanshu(x2)
    IF sqn(y1)=sqn(yz) THEN
      PRINT "NO ROOT."
    ELSE
    DO
        LET x=(x1+x2)/2
        LET y=hanshu(x)
        IF SGN(y1)=sgn(y) THEN LET x1=x ELSE LET x2=x
    LOOP until abs(x1-x2)<e
    PRINT"ROOT:"; (x1+x2)/2
    END IF
END SUB
INPUT PROMPT "a,b,eo=? :a,b,eo
CALL binary(a,b,eo)
END
Ok.RUN
a,b,eo=? 1,2,0.0001
ROOT:1.47568
OK.RUN
a,b,eo=? 2,3,0.0001
NO ROOT
```

习 题 七

1. 用标准函数计算：

(1)根据三角函数公式

SIN(A+B)=SIN(A)COS(B)+SIN(B)COS(A)

编一个程序验证等号两边相等。

取数据：A=10°,20°,30°,40°
B=20°,30°,40°,50°

(2)已知A(1,3),B(6,2),用公式R=$\frac{1}{2}$absinr,

计算三角形ABO的面积，见图7-11所示。

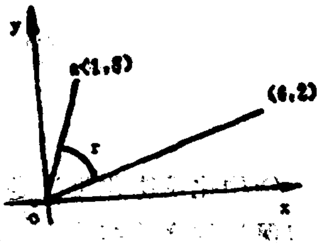

图 7-11

2.判断下列程序的执行结果。

```
(1)def F(c)=32+9*c/5
    INPUT    C
    PRINT "F="; F(C)
    END
    (2)PRINT"$$$$1"
    CALL  A
    PRINT"××××2"
    CALL  B
    PRINT"####3"
    END
    SUB A
    PRINT"@@@@"
    END SUB
    SUB B
    PRINT"****"
    END SUB
```

```
(3)FOR  I=1  TO 5
   Call  ABC
   NEXT I
   END
   SUB ABC
   PRINT I
   END SUB
(4)LET X=3
   DETA(W)=W*W*5*W+5
   LET B=2*X+A(3*X)
   PRINT B
   END
```

3.用自定义函数定义

$$z=5.16x+\ln(x+\sqrt{x}+\frac{1}{x})$$

求(1)　y=2.0时

$$H=\sin(18,y)+15.48y+\ln(3y+\sqrt{3y}+\frac{1}{3y})$$

(2)A=3.6,B=2.1,c=5.8时

$$G=\frac{5.16(A+B)+\ln(A+B+\sqrt{A+B}+\frac{1}{A+B})}{5.16c+\ln(c+\sqrt{c}+\frac{1}{c})}$$

4.编写一程序，求半径分别为1，2，3，4，5的圆面积。

5.当X=3时分别求$4X^2+\sqrt{X+1}$，$4X^2+\sqrt{X+1}+10$及$2(2X^2+\sqrt{X+1})\cdot(4X^2+\sqrt{X+1}+10)$的值。

6.求出三组随机数，每组三个数，用子程序输出这三组数，用子程序找出每组中最大的数。

7.试将下列单词按英文字母顺序排列出来。

bead.about,capaciter,damp,fail,beep,impedance,meet,oil,power,setting,daker.

8.有自然数1，2，3…n(n<100) 除1和2之外其余的自然数之和是100的整倍数。试求n=？

（提示）：3+4+5+…+n=(n-2)(n+3)/2

当(n-2)(n+3)/2的值为100的整数倍时n就是所求的值。

9.将N个数中的负数选出来，使用子程序将它们置入一维数组，再用一个子程序使它们按升序（从小到大）排列输出。

第八章　结构化程序设计

结构化模块化设计是保证程序质量的基础。True BASIC是一种结构化程序设计系统，可以保证程序的结构化。结构化可使程序有严密的逻辑性，易于进行程序的检错和测试验证；模块化设计可将一个大的复杂的程序化整为零，分割成小的模块，这样既便于程序设计简化，又能提高程序的生产效率，使用积木式组合技术。

True BASIC实现结构化程序设计的主要方法有：

1.子程序：以SUB开始，END SUB结束的子程序提供了把大程序化为小模块的方法。

2.自定义函数：以DEF开始，END DEF结束的自定义函数提供了除标准函数以外把大程序划分为小段的又一种方法。

3.库文件：以EXTERNAL开始的库文件，是存放在外部介质中的函数和子程序集合，库文件中的函数和子程序可以方便地被任何程序调用。

True BASIC程序以END为界，将模块分为内部模块和外部模块。当自定义函数及子程序位于END语句前边时，是所属程序的内部模块，又叫内部函数和内部子程序。位于END语句后边，则叫外部函数和外部子程序，是所属程序的外部模块。第七章所讲的全部是内部函数和内部子程序，本章介绍外部函数和外部子程序。外部模块与内部模块的主要区别是变量的所属关系不同。

§8-1　整体变量和局部变量

请看下面的程序及运行结果：

```
! 程序A
SUB sub1
    PRINT i,
END SUB
FOR i=1 TO 5
    CALL sub1
NEXT i
END
OK.RUN

1       2       3       4       5
```

```
! 程序B
FOR i=1 TO 5
    CALL sub1
NEXT i
PRINT
END
SUB sub1
    PRINT i,
END SUB
OK.RUN

0       0       0       0       0
```

程序A中，五次调用子程序sub1，输出的i值与循环变量i的值相同，这说明位于END语句前边的子程序sub1中的i与循环变量i占用同一个内存单元。

程序B中，五次调用子程序sub1，输出的i值都是0，说明i没有被赋值，这说明位于END语句后的子程序sub1中的i与循环变量i不占用同一个内存单元。

再比较下面两个程序：

第一个程序把cheng (x) 定义为内部函数，第二个程序把cheng (x) 定义为外部函数。

```
! 第一个程序
DEF cheng(x)
   LET c=0
   FOR i=1 TO n
      LET c=c+x*x
   NEXT i
   LET cheng=c
END DEF
LET n=4
LET t=cheng(5)
PRINT"t="; t
END
OK.RUN
t=100
```

```
! 第二个程序
DECLARE DEF cheng
LET n=4
LET t=cheng(5)
PRINT"t="; t
END
DEF cheng(x)
   LET c=0
   FOR i=1 TO n
      LET c=c+x*x
   NEXT i
   LET cheng=c
END DEF
OK.RUN
t=0
```

请读者仔细分析，为什么这两个程序运行的结果不同。

一、整体变量

整体变量又叫全局变量，即变量在整个程序进行过程中都是有效的，同名变量占用同一个内存单元，内部函数和内部子程序与主程序的变量公用内存单元，是整体变量。如程序A中的sub1子程序中的i与主程序中的i。在BASIC语言中所有的变量都是全局变量。由于这一特点，在大型程序设计中，极易造成程序模块间的相互干扰，从而带来极为不利的影响。

为了克服BASIC语言这一严重的弊端，在True BASIC中引入了局部变量概念。

二、局部变量

局部变量指程序模块中的变量单元分配与其它程序的变量单元分配无关，即使名字完全相同，也不公用内存单元，所以各模块之间变量的命名、数据的变化没有直接关系，互相独立，外部函数和外部子程序中的变量仅在其模块内部有效。所以，在程序B中，调用外部子程序sub1时，子程序中的i与主程序中的循环变量i无关。

局部变量的引入使程序模块可以单独编辑和调试，编辑程序模块时，可以方便地选择变量，不必担心因为和其它模块有同名变量导致错误；同时，调用各功能模块时，也无需了解模块内部的细节。所以，外部模块是结构化程序设计的基础。为了正确编辑和调用程序模块，必须从局部变量和整体变量的观点分析和认识各种模块的结构和功能，以及内部模块与外部模块的差异。

注意：外部模块被调用前，所有变量都被置零或空字符串。调用时，调用它的程序单位通过形式参数把实际参数传给外部模块的变量。调用后，外部模块变量的值通过形式参数传给调用它的程序单位。外部模块只要返回到调用它的程序单位，所有的变量值都不能被保留。

§8-2 外部模块

外部模块指外部函数、外部子程序及其集合——库文件。

一、外部函数

定义方式和调用规则与内部函数基本相同，请参阅第七章。

1.外部函数特点

(1) 外部函数被定义在主程序的END语句之后或其他外部子程序、外部函数的外面，它是一个独立的程序单位。

(2) 外部函数的所有变量都是局部变量，即只对该外部函数有效，它与其他程序单位没有共享变量。

(3) 其他程序单位使用外部函数之前，必须用DECLARE DEF语句进行说明。此语句格式为：

DECLARE DEF〈函数名〉,〈函数名〉, …

其作用是通知True BASIC系统已被定义，所以，可以与置于调用语句后的内部函数用同一个命名语句。

(4) 外部函数的定义与内部函数一样，也有单行函数和多行函数两种形式。

(5) 外部函数和内部函数一样，也可以有参数，也可以没有参数。对于多行函数也要用赋值语句，即 LET 语句，把计算的函数的实际值赋给函数名，也可以包含一个或多个EXIT DEF语句，当程序执行到该语句时，便返回调用它的程序处。

(6) 函数返回的值必须是数值与字符串，若返回一个字符串值，则函数名必须以“$”结尾。

(7) 其他程序单位每次调用外部函数时，函数都收到一份全部（局部于该外部函数）的重新初始化的副本，即它的所有变量初始化为零或空字符串。故外部函数可以递归调用自己。

(8) 和内部函数一样，函数名也是程序中的一个整体变量，故不能与主程序的其它变量同名。

2.应用举例

例1 当N>2时，先计算

$$s1=\sqrt{1}+\sqrt{2}+\cdots+\sqrt{N}$$

然后再计算

$$s=s1+\sqrt{N+1}+\sqrt{N+2}+\cdots+\sqrt{INT(s1)}$$

求s1和s的过程相似，都是求

$$\sqrt{M}+\sqrt{M+1}+\cdots+\sqrt{MM}$$

只是求s1时，初值M=1，终值MM=N；

求S时，初值M=N+1，终值MM=INT (s1) 。

程序及运行结果为：

```
DECLARE DEF sum                    ! 外部函数名定义
READ n
```

```
LET s=sum(1,n)+sum(n+1,int(sum(1,n)))
PRINT“s=”; s
DATA 8
END
DEF sum(m,mm)                          ! 外部函数定义
    LET s=0
    FOR i=m TO mm
        LET s=s+SQR(i)
    NEXT i
    LET sum=s
END DEF
OK.RUN
s=44.4692
```

该程序中的外部函数和主程序都用到变量s，但它们却有不同的意义和不同的值。外部函数中的s在进入主程序后无效，所以用 LET sum=s，通过函数名sum把s值回带到主程序中来。主程序中的第五行赋值号右部的表达式三次调用了函数sum。

例2 计算n!，定义一个递归调用的外部函数fact供主程序调用。

程序及运行结果为：

```
DECLARE DEF fact
FOR i=1 to 5
   PRINT“i=1”; i,“i! =”; fact(i)
NEXT i
END
DEF fact(n)
     IF n<=1  THEN LET fact=1 ELSE LET fact=n*fact(n-1)
END DEF
   OK.RUN
   i=1          i! =1
   i=2          i! =2
   i=3          i! =6
   i=4          i! =24
   i=5          i! =120
```

外部函数中用数组作参数时，同内部函数一样，函数的形参须标明维数：

〈数组名〉(　　) 一维数组

〈数组名〉(,　) 二维数组

例如a(, ,)，a是三维数组。

例3 统计学生成绩，主程序中读入学生的学号numb，及三科成绩a(1)，a(2)，a(3)，调用外部函数，计算三科平均成绩，若有缺考时，用－1表示，其平均成绩，直接赋－1，并用EXIT DEF退出外部函数，返回主程序。

程序及运行结果为：

```
DECLARE DEF aver
DIM test(3)
```

```
PRINT"学号", " 成 ", " ", " 绩      ", " 平均成绩"
PRINT"…………………………………………………"
DO WHILE MORE DATA
  READ numb
  PRINT numb,
  FOR i=1 TO 3
       READ test(i)
       PRINT tesr(i),
  NEXT i
  LET sav=aver(test,3)
  PRINT sav
LOOP
DATA 1111,89,76,90,1112,80,-1,98,1113,79,85,40,1114,-1,78,88
DATA 1115,98,89,99,1116,86,91,81,1117,67,86,88,1118,98,79,86
END
DEF aver(a(  ),n)
  LET ar=0
  FOR i=1 TO n
       IF a(i)<0 THEN
         LET aver=-1
         EXIT DEF
       END IF
       LET ar=ar+a(i)
   NEXT i
   LET aver=ar/n
END DEF
OK.RUN
```

```
学号          成            绩              平均成绩
………………………………………………………………………

1111          89          76          90          85
1112          80          -1          98          -1
1113          79          85          40          68
1114          -1          78          88          -1
1115          98          89          99          95.3333
1116          86          91          81          86
1117          67          86          88          80.3333
1118          98          79          86          87.6667
```

例4　由20个数组成的裴波那契数列，它的头两个数为0和1，第三个数是第一、第二个数之和，第四个是第二、三数之和，以后每个数都是其前两个数之和，也就是说这个数列为：

0，1，1，2，3，5，8，…

试输出这二十个数，并求出这二十个数的平均值R，设R_i表示这些数，计算它们的方差D和

标准差 s, $D=\frac{1}{20}\sum_{i=0}^{19}(R_i-R)^2$, $s=\sqrt{\frac{1}{20}\sum_{i=0}^{19}(R_i-R)^2}=\sqrt{D}$

裴波那契数列可描述为裴波那契数：

$$f(n)=\begin{cases}0, & 当n=0\\ 1, & 当n=1\\ f(n-1)+f(n-2) & 当n>0\end{cases}$$

因此，我们可以把上述函数定义为一个递归函数。并把这二十个数存放在一个一维数组中，然后，我们再定义一个外部函数，其功能是求一维数组所有元素的平均值：

程序及运行结果为：

```
DECLARE DEF fibna,rv
DIM a(0 to 19)
FOR j=0 TO 19
   LET ro=fibna(j)
   LET a(j)=ro
NEXT j
MAT PRINT a;
LET r=rv(a)
PRINT"R=""; r
LET so=0
FOR i=0 TO 19
   LET ri=a(i)-r
   LET SO=SO+ri∧2
NEXT i
LET c=so/20
LET s=sqr(d)
PRINT"D="; d
PRINT"s="; s
END
DEF fibna(n)
    IF n=0 THEN
        LET fibna=0
    ELSE
        IF n=1 THEN
           LET fibna=1
        ELSE
           LET fibna=fibna(n-1)+fibna(n-2)
    END IF
    END IF
END DEF
DEF rv(b())
   LET s=0
```

```
    FOR i=lbound(b)  TO  ubound(b)
        LET s=s+b(i)
    NEXT i
    LET rv=s/size(b)
END DEF
OK.RUN
 0  1  1  2  3  5  8  13  21  34  55  89  144  233  377  610  987  1597  2584  4181
R=547.25
D=1.1147e+6
s=1055.81
```

二、外部子程序

外部子程序的定义形式及调用规则与内部子程序基本相同，请参阅第七章。

1.外部子程序特点

(1) 外部子程序是一个独立的程序单位。书写在主程序的END语句之后。

(2) 外部子程序中所有变量都是局部变量。仅用于该外部子程序。它完全通过虚实结合与主程序传送数据。

(3) 外部子程序与外部函数不同，在调用之前不需要作任何说明。

(4) 外部子程序可以有参数，也可以没有参数。

(5) 外部子程序中也可以包含EXIT SUB语句，执行此语句后，立即退出子程序而返回到调用它的主程序处。

(6) 外部子程序名也必须是合法的变量名，不能以“$”作为子程序名的结尾。

2.应用举例

例5 用边公式计算三角形面积。当给的边长不能组成三角形时，面积给出—1标志，直接输出边长和—1标志。

第一种编程方法。面积直接在计算面积的子程序中输出，三边之和的二分之一值用函数sp计算。程序流程图见图8-1。

程序及运行结果为：

```
! 外部函数及外部子程序
DECLARE DEF sp
PRINT"A","B","C","Area"
PRINT"........................................................"
DO UNTIL END DATA
    READ a,b,c
    CALL area(a,b,c)
LOOP
DATA 1,2,3,2,3,1,3,1,2,1,1,1,3,4,5
END
DEF sp(a,b,c)                                ! 外部函数
    IF a+b<=c OR ABS(a-b)>=c THEN
        LET sp=-1
        EXIT DEF                             ! 跳出函数
```

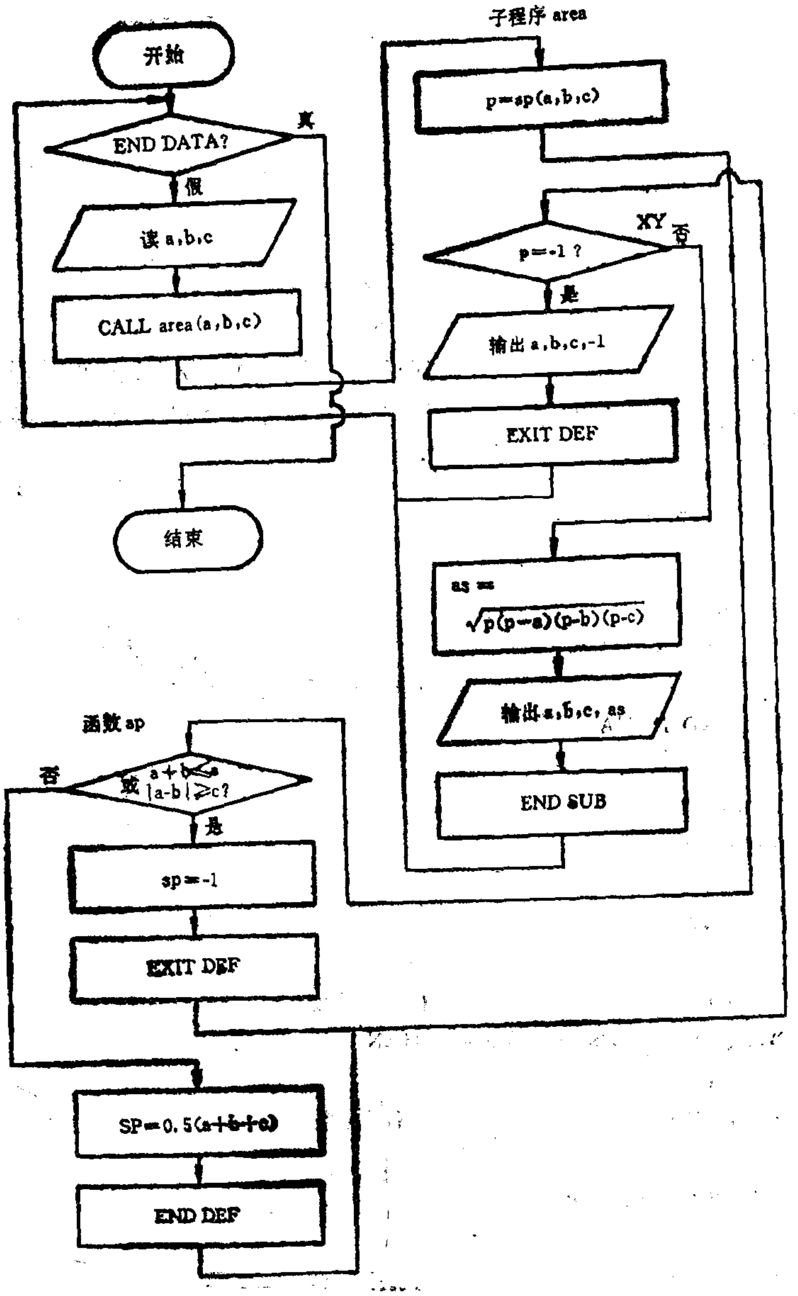
子程序 area
开始
END DATA?
真
假
读 a,b,c
CALL area(a,b,c)
结束
p=sp(a,b,c)
p=-1 ?
否
是
输出 a,b,c,-1
EXIT DEF
as = √p(p-a)(p-b)(p-c)
输出 a,b,c, as
END SUB
函数 sp
a+b≤c 或 |a-b|≥c?
否
是
sp=-1
EXIT DEF
SP=0.5(a+b+c)
END DEF

图 8-1

```
    END IF
    LET sp=0.5*(a+b+c)
END DEF
SUB arda(a,b,c)                    ! 外部子程序
    DECLARE DEF sp                 ! 函数命名
    LET p=sp(a,b,c)
    IF p=-1 THEN
       PRINT a,b,c,-1
       EXIT SUB                    ! 跳出子程序
    END IF
    LET as=SQR(p*(p-a)*(p-b)*(p-c))
```

```
      PRINT a,b,c,as
    END SUB
OK.RUN
    A              B              C              Area
    ..........................................................
    1              2              3              -1
    2              3              1              -1
    3              1              2              -1
    1              1              1              .433013
    3              4              5              6
```

第二种编程方法：输入、输出语句都设在主程序中，用参数把子程序中计算的面积值带回主程序中再输出。调用子程序时，这个用于带回面积值的参数必须是主程序中的一个变量，不能用表达式。

程序及运行结果为：

```
  ! 外部子程序
  DECLARE DEF sp
  PRINT "A","B","C","Area"
  PRINT"..........................................................."
  DO UNTIL END DATA
      READ a,b,c
      CALL area(s,a,b,c)
      PRINT a,b,c,s
  LOOP
  DATA 1,2,3,2,3,1,3,1,2,1,1,1,3,4,5
  END
  DEF sp(a,b,c)                          ! 外部函数
      IF a+b<=C OR ABS(a-b)>=c THEN
         LET sp=-1
         EXIT DEF                        ! 跳出函数
      END IF
      LET SP=0.5*(a+b+c)
 END DEF
 SUB area(as,a,b,c)                      ! 外部子程序
     DECLARE DEF sp                      ! 函数命名
     LET p=sp(a,b,c)
     IF P=-1 THEN
        LET as=-1
        EXIT SUB                         ! 跳出子程序
     END IF
     LET as=SQR(P*(p-a)*(p-b)*(p-c))
 END SUB
 OK.RUN
     A              B              C              Area
     ..........................................................
         1          2              3              -1
```

```
2          3          1          -1
3          1          2          -1
1          1          1           .433013
3          4          5           6
```

例6 有m行m列的数组a，计算各行元素之和，存入一维数组s中。编一个程序，读入4行4列数组a，调用外部子程序suba，按行求元素之和存入数组s。并输出数组a和s的全部元素。

子程序：

```
SUB  suba (ab (, ), s ( ), m)
```

参数　ab(,) 是二维数组
　　　s ( ) 是一维数组
　　　m　　 是一个数字型参数

调用语句：

```
CALL suba (a, s, 4)
```

a和s必须与形参ab和s的维数相同，调用语句中的实参不必写入具体维数和下标个数。

程序及运行结果为：

```
DIM a(4,4), s(4)
FOR i=1 TO 4
   FOR j=1 TO 4
      READ a(i, j)
      PRINT a(i, j),
   NEXT j
   PRINT
NEXT i
CALL suba(a, s, 4)
PRINT"          数组s:"
PRINT"........................................."
FOR k=1 TO 4
   PRINT s(k),
NEXT k
PRINT
DATA 12, 34, 33, 45, 25, 40, 24, 45, 9, 10, 12, 34, 45, 34, 10, 23
END
SUB suba(ab(, ), s(  ), m)
   FOR i=1 TO m
   LET sa=0
   FOR j=1 TO m
      LET sa=sa+ab(i, j)
   NEXT j
   LETS(i)=sa
NEXT i
END SUB
 OK,RUN
```

```
12           34           33           45
25           40           24           45
9            10           12           34
45           34           10           23
                       数组 s:
.............................................................
124          134          65           112
```

例7 给定一个一维数组a(10)，其中存放十个数，编一排序的程序。在这个程序中，定义一个内部子程序Sort1，其功能是将这十个数按由小到大顺序排序，再定义一个外部子程序Sort2，其功能是将这十个数按由大到小排序：

```
! Sorting
SUB Sortl(e(  ))
  FOR i=lbound(c)TO ubound(c)-1
    FOR j=i+1 TO ubound(c)
        IF c(i)>c(j)THEN
            LET x=c(i)
            LET c(i)=c(j)
            LET c(j)=x
        END IF
  NEXT j
  NEXT i
MAT PRINT c;
END SUB
DIM a(10), b(10)
MAT INPUT a
MAT b=a
CALL Sortl(a)
CALL Sort2(b)
END
SUB Sortz(e(  ))
    FOR i=lbound(c) TO ubound(c)-1
    FOR j=i+1 TO ubound(c)
      IF c(i)<c(j)THEN
        LET x=c(i)
        LET c(i)=c(j)
        LET c(j)=x
    END IF
  NEXT j
NEXT i
MAT PRINT c;
END SUB
OK.RUN
? 12,·-7, 23, -15, 54, 10, -25, 35, -8.5
-25 -15 -8 -7 5 10 12 23 35 54
54 35 23 12 10 5 -7 -8 -15 -25
```

三、库文件

为了便于开发大型应用软件，True BASIC提供了库文件的建立和使用方法。所谓库文件是一些外部函数、子程序的集合，它集中一些具有一定功能的子程序或函数于一个外部文件中，其中没有主程序，但在任何程序中均可以作为外部模块去调用它们。所以库文件就是一个软件包，可供设计程序时调用，以节省开发时间。

1.库文件的建立

(1) 编辑形式。第一语句是EXTERNAL。

在EXTERNAL 语句之后是外部函数、外部子程序的顺序排列，它们的顺序是任意的。但是应注意，在同一个库文件中，函数或子程序名不能相同，也不能与调用程序单位中的变量、函数或子程序名相同。

库文件可以是编译的，也可以不被编译，一般常用的库文件是编译后再用，则更方便。

(2) 存储方式。用一个文件名把EXTERNAL开头的含有几个无主程序的外部函数和外部子程序的库文件作为整体，使用SAVE命令将它们存放在外部介质上（如磁盘上）。

例如，下面是两个用EXTERNAL开始的库文件，存盘的文件名分别是ku-1和 ku-2，其中都含多个子程序或函数。

ku 1.TRU文件:

```
EXTERNAL
DEF sp(a, b, c)                          ! 外部函数
    IF a+b<=c OR ABS(a-b)>=c THEN
        LET sp=-1
        EXIT DEF                          ! 跳出函数
    END IF
    LET sp=0.5*(a+b+c)
END DEF
!
SUB arca(as, a, b, c, p)
    LET as=-1
    IF P<>-1 THEN
        LET as=SQR(p*(p-a)*(p-b)*(p-c))
    END IF
END SUB
!
DEF ab(a, b)
    IF a>b THEN LET ab=a ELSE LET ab=b
END DEF
!
SUB ppp(k, )
    FOR i=1 TO k
        PRINT "*";
    NEXT i
PRINT
END SUB
```

ku—2.TRU文件：

```
EXTERNAL
SUB abc(a, b, c, s)
   LET S=a+b+c
END SUB
SUB ppp(k, )
   FOR i=1 TO k
      PRINT "*—";
   NEXT i
   PRINT
END SUB
DEF u(a)
   LET u=SQR(a)
END DEF
```

然后用SAVE命令将它们存入磁盘，形式是：

```
SAVE〈文件名〉
```

2.使用库文件

当程序要调用库文件中的某一个或某几个函数或子程序时，应在调用之前使用一条LIBRARY语句，并指出要检索的库文件名。形式为：

```
LIBRARY "〈文件名1〉", "〈文件名2〉", …
```

这样就可以用通常方式调用库文件中的函数和子程序。在实现比较复杂的算法时，利用库文件的优越性是明显的。

在这里，〈文件名〉是指库文件存盘时用的文件名。

例8 调用上述两个库文件中的某些函数和子程序，进行如下操作：

读入三角形边长，判别能否构成三角形，如果能，计算三角形面积；否则给面积赋值－1；输出边长和面积；打印“*”，其个数等于三边之和的平方根。最后找出最大的三角形面积值。

程序及运行结果：

```
LIBRARY"a:ku—2", "a:ku-1"                              ! 说明库文件名
DECLARE DEF u, sp                                      ! 说明外部函数名
FOR i=1 TO 4
    READ x, y, z
    LET p=sp(x, y, z)
    CALL area(s, x, y, z, p)
    PRINT x, y, z, s
    CALL abc(x, y, z, aa)
    LET dsp=u(aa)
    PRINT TAB(50);
    CALL ppp(dsp)
    LET w=w+aa
NEXT i
PRINT
PRINT"w="; w
DATA 9, 5, 3, 2, 4, 5, 3, 1, 2, 4, 5, 6
```

```
 END
 OK.RUN
9          5          3          -1
                                  *—*—*—*—
2          4          5          3.79967
                                  *—*—*—
3          1          2          -1
                                  *—*—
4          5          6          9.92157
w=49                              *—*—*—
```

LIBRARY语句的作用是通知True BASIC系统检索库文件，将LIBRARY指定的全部外部函数和子程序组合在一起，而在使用上应和程序中的外部函数和外部子程序相同。

实际使用的过程是比较复杂的。首先将所有程序单位填入当前文件，按出现的先后顺序列出所给库文件表。然后检索库文件，寻找尚未定义的函数或子程序。打开文件表头上的文件目录，从文件目录中取出所需要的程序。

所以，使用库文件，可以存入常用算法程序，简化程序设计，节省时间，实现软件资源共享。但由于步骤繁多速度也受到一定影响。

3.系统库文件

在True BASIC系统中有几个函数和子程序库文件，其中有四个包含数学函数库，一个包含画图子程序库；一个包含使用菜单子程序库。

§8-3 程序的分解与链接

前面介绍的主程序、函数、子程序均属于一个程序文件中，在运行时一次装入内存。有时当程序较大，需要占用较多的内存时，可以将一个大程序分割为几个独立程序块，可以一个接一个地执行这些程序块，执行一个程序块时，另一个程序块所占的内存被释放，这种方式称为覆盖方式执行程序。由一个程序转向执行另一个独立程序文件的方式称为链接。

一、CHAIN语句

当一个CHAIN语句被执行时，当前程序被停止。转而开始执行它指名的程序。所有当前程序所使用的存储单元都被释放。所以，它可以链接在内容上没有直接关系的程序，使之一个接一个地被执行，就象在DOS系统下执行一个批处理文件。

例9 有三个互不相关的程序，分别用A，B，C存入磁盘（扩展名.TRU）。

程序及运行结果为：

A.TRU程序及运行结果为：

```
 LET ABC$="A----A----A"
 PRINT ABC$
 LET ABC=555
PRINT"A="; ABC
END
OK.RUN
```

```
A----A----A
    A=555
```

B.TRU程序及运行结果为：

```
LET ABC$="B---B---B"
PRINT ABC$
LET ABC=111
PRINT"B="; ABC
END
OK.RUN
B---B---B
B=111
```

C.TRU程序及运行结果为：

```
LET ABC$="C---C---C"
PRINT ABC$
LET ABC=666
PRINT"C="; ABC
END
OK.RUN
C----C----C
    C=666
```

若希望按A→B→C顺序逐个程序连续运行：为此目的，有两种方式，用CHAIN把这三个程序链接，使之按规定顺序执行。

1.单向链接

CHAIN "〈文件名〉"

文件名是程序存盘时用的文件名，是链接语句的目标文件。执行这个语句时，首先释放内存单元（原文件被清除）再把目标文件装入内存并运行。

为能按顺序执行A，B，C三个不同的程序模块，将A和B文件作如下修改：

在B文件的END语句前插入语句：

CHAIN "C"

并将修改后的该程序改名B-B存入磁盘。这样c文件就是B-B文件的链接的目标文件。当运行B-B文件时，运行到CHAIN"C"时，释放内存中的B-B程序，装入C文件并运行之，所以等价于运行B和C两个程序。

B-B.Tru程序及运行结果为：

```
  LET ABC$="B----B----B"
  PRINT ABC$
  LET ABC=111
  PRINT#1: "B="; ABC
  CHAIN  "C"
END
OK.RUN
B----B----B
```

```
B=111
C---C----C
C=66666
```

如果把B-B又作为A文件的目标文件被链接，A改为下面A-A程序，运行A-A，可实现连续依次运行AA→BB→C，等价于把A-B-C用单链串在一起运行。

A-A.TRU源程序及运行结果为：

```
LET ABC$=“A……A……A
PRINT ABC$
LET ABC=555
PRINT“A=”; ABC
CHAIN “B-B”
END
OK.RUN
A…A…A
A=555
B---B---B
B=111
C---C---C
C=66666
```

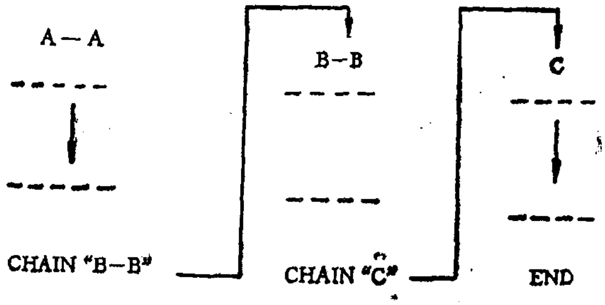

图 8-2

链接示意图见图8-2所示。

注意：这种链接方式，内存中总是存储当前运行的一个程序。

2.返回链接

CHAIN “〈文件名〉”，RETURN

这个语句被执行时，不释放内存的当前程序。直接进入目标文件执行，直到目标文件的END语句又返回原程序的本次CHAIN的下一个语句，继续执行原来的程序。

例10 程序ABc-3c.Tru的程序及运行结果为：

```
LET ABC$=“ABC----ABC----ABC”
PRINT ABC$
LET ABC=3355511666
PRINT ABC
PRINT“链接运行开始：”
CHAIN “A”，RETURN
CHAIN “B”，RETURN
CHAIN “C”，RETURN
PRINT “ALL END”
END
OK.RUN
ABC----ABC----ABC
3355511666
链接运行开始：
A----A----A
A=555
B----B----B
```

```
B=111
C---C----C
C=666
ALL END
```

链接关系见图8-3所示。

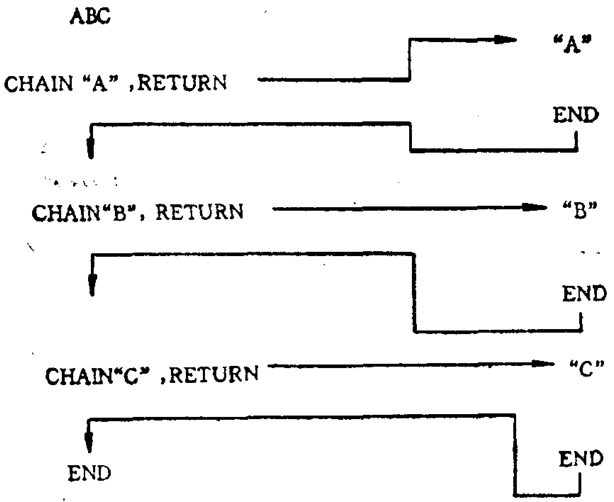
ABC
CHAIN "A" ,RETURN
"A"
END
CHAIN"B", RETURN
"B"
END
CHAIN"C" ,RETURN
"C"
END
END

图 8-3

二、PROGRAM语句和有参链接

1.PROGRAM语句

格式： PROGRAM 〈程序名〉(〈形参〉)

程序名： 是任一个合法的变量名，与链接时所使用的存盘文件名无关。

形参： 是一个合法的字符串变量名，因为只能传递字符串，它是一个字符型的形参。

功能： 说明程序被链接时传递的参数，当程序被链接时，程序中的形参将被链接语句中实参所置换。如果没有形参，则PROGRAM 语句可以缺省，即使有也无任何操作，是非执行语句。

2.有参链接

有参链接也分单向链接和返回链接两种方式。在进行有参链接时，目标程序的第一个语句必须是PROGRAM语句，用以说明被传送的参数。

例11 用P1作为文件名存盘的P1.Tru源程序：

```
! P1
PROGRAM a3(s$)
OPEN #2: PRINTER                        ! 打开打印机通道
PRINT #2: "-----------------"
PRINT #2:s$
PRINT #2: "-----------------"
END
OK.RUN
----------------------
----------------------
```

程序中的a3是程序名，有一个参数s$，程序中有一个输出s$的语句。如果直接运行这

个程序，将输出一个空格；a3作目标被链接时，输出实参的值。

以a3作目标程序，用如下三种链接方式，将得到三种运行结果。

(1) 没有RETURN链接，程序不能运行完整。CHAIN语句以后的部分不能被执行。

程序及运行结果：

```
! P1a
OPEN #1:PRINTER                              ! 打开打印机通道
PRINT #: "A"
LET p$= "**    **    **    **    **    **"
CHAIN "p1" WITH(p$)
PRINT #1: "*A*"
END
OK·RUN
A
------------------------------------
**    **    **    **    **    **
```

(2) 有RETURN链接，但程序中打印机通道不关闭，结果先执行目标程序的输出，再执行程序的输出。

程序及运行结果为：

```
! P
OPEN #1: PRINTER                                ! 打开打印机通道
PRINT #1: "A"
LET p$= "**    **    **    **    **    **"
CHAIN "p1" WITH(p$), RETURN
PRINT #1: "*A*"
END
OK·RUN
---------------------------------------------
**    **    **    **    **    **
---------------------------------------------
A
*A*
```

(3) 先关闭原文件的打印机通道，再链接目标程序，返回原文件后再重新打开打印机。结果输出顺序是：原文件→目标文件→原文件，符合输出语句的先后顺序。

程序及运行结果为：

```
! Pa
OPEN #1: PRINTER                             ! 打开打印机通道
PRINT #1: "A"
LET p$= "**    **    **    **    **    **"
CLOSE #1
CHAIN "p1" WITH(p$), RETURN
OPEN #5: PRINTER
```

```
PRINT #5: "*A*"
END
OK·RUN
A
-------------------------------------------
**    **    **    **    **    **
-------------------------------------------
*A*
```

习 题 八

1. 复习、总结、归纳：整体变量、局部变量的相同点和不同点；标准函数、内部函数、内部子程序、外部函数、外部子程序、库文件，有哪些相同点和不同点。

2. 如何建立和使用库文件？

3. 编写两个矩阵相乘的子程序，并用来计算

A·B=C，D·E=F

各矩阵大小为：A:7×5　　B:5×3　　D:4×6　　E:6×6

4. 编写一个计算n次多项式

$$Y=\sum_{i=0}^{n} a_i x^{n-i}$$

的子程序，并用来计算

$$F=\sum_{i=0}^{100} bri(u^2+v^2)^{100-i}$$

5. 见8-4图所示。其中a:b:c=11:5:2

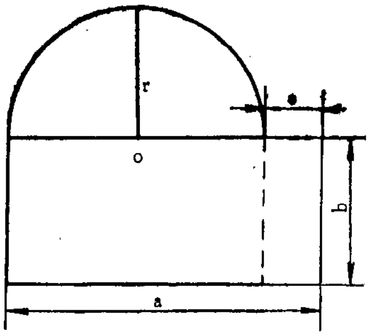

图 8-4

(1) 把该图形分成四部分，分别用不同形式内部、外部、函数、子程序的结构模块计算面积，用一个主程序调用各模块，计算总面积。

(2) 把各模块分别装入二个库文件中，用调用库文件的方式计算图形总面积。

(3) 把计算四部分面积的程序编成独立源程序，用CHAIN链接，分步计算、输出各部分图形的面积。

第九章 字符处理

在计算机应用的许多方面，如检索、编辑、文字信息处理等，都和字符型数据运算有关。标准函数中的字符串函数也是用于字符型数据运算的。本章将着重讨论字符型数据的运算与处理问题。

§9-1 字符型变量的运算

一、字符串的连接与字符串表达式

将两个或多个字符型变量的值相互连接在一起组成一个新的字符串，这种运算称为连接，并以“&”作为连接的运算符。

例如：
```
LET A$= "abcde"
LET B$= "fghij"
LET C$=A$ & B$
PRINT C$
```

在连接运算中，A$在前，B$在后。连接的结果组成字符串C$，其值是“abcdefghij”，即将A$串尾与B$串首连接在一起。

由字符常量、字符变量或字符串数组元素以及字符串函数（取值为字符型）之间用连接字符“&”相连接组成的表达式，称为字符串表达式。字符串表达式的值仍是字符串。

如上所述，A$&“   ”&B$也是一个字符串表达式，其值为“abcde fghij”。

例1 这里给出一段程序，将输入的姓名和地址连接成一个字符串，其间插入三个空格。若组合串不超过70个字符，则在一行上输出，否则分作两行，输出姓名和地址。

```
! 输出姓名和地址
SET MODE "HIRES"
INPUT PROMPT "Enter your name: " : name$
LINE INPUT PROMPT "Address: " : address$
LET a$=name$& "   " &address$
IF len(a$)<70 THEN
   PRINT a$
ELSE
   PRINT name$
   PRINT address$
END IF
END
```

二、字符串的比较

两个字符串可以进行比较，以确定它们之间的大小。比较是由各串最左端的第一个字符

开始，依次逐个比较每个字符，如果两个字符串长度相同，且对应位置的每个字符相同，则两个字符串相等。若长度不等，或对应位置上字符不同时，这两个字符串就不相等。

对两个不等字符串比较，有两种情况：如果前部各对应位置上字符均相同，但字符串长度不同，则字符个数多的串为大（如：“Books”＞“Book”）；如果在逐个字符比较时，出现了不相同的字符时，则比较这两个字符的大小，以字符的ASCII码值较大的字符串为大（ASCII码见附录一）。

如字母A的ASCII码为65，字母B的ASCII码为66，显然有“B”＞“A”，“B”＞“AB”；有下述关系表达式均为真：“ACB”＞“ABC”，“ADC”＞“ABC”，“1234”＜“1236”，“123”＞“12”等。

例2 现有用英文表示的一批国家名，要求找出按字典顺序排列在最前面的国家。所谓字典顺序，实际上就是字符串的从小到大的顺序，排列在最前面的是最小的。

```
! 找出最前面的国名
  READ a$
  FOR I=1 TO 7
      READ B$
      IF B$<A$ THEN LET A$=B$
  NEXT I
  PRINT A$
  DATA "CHINA", "JAPAN", "CANADA", "KOREA"
  DATA "ENGLAND", "FRANCE", "AMERICA", "INDIA"
  END
 OK.RUN
  AMERICA
```

三、取子字符串

当给定了一个字符串后，在该字符串中，取出其中连续一部分字符组成的字符串，称为该字符串的子字符串，简称子串。

如给定A$＝“ABCDE”，则“A”，“AB”，“CD”，“CDE”，“BCD”等均是A$的子串，而“AC”，“BDE”，“BA”，“EF”等均不是A$的子串。

如字符串变量名为A$，则A$（n:m）表示取A$中由第n个字符始至第m个字符止的子串。

例如，A$＝“ABCDE”，A$（2:3）＝“BC”，A$（1:1）＝“A”。

例3 编一段程序，将一英语句子分解成单词输出。设句子长度不超过80个字符，句中无任何标点符号，单词之间用空格分隔。

设用两个字符型变量，Text$保存英语句子，Word$保存每次分解出来的单词。

```
! 单词输出
LET word$=" "
INPUT text$
LET text$=text$&" "
LET k=1
FOR  i=1  TO  80
```

```
IF text$(i:i)<>" "THEN
  LET word$(k:k)=text$(i:i)
  LET k=k+1
ELSE
  IF k<>1 THEN
    PRINT word$
    LET k=1
    LET word$=" "
  END IF
END IF
NEXT i
END
OK.RUN
? The computer can read english
The
computer
can
read
English
```

例4 试分析下面程序及其运行结果为：

```
LET string$= "Tian Anmen"
PRINT string$
LET string$(1:0)= "Beijing"
PRINT string$
LET string$(Maxnum:0)= "Guangchang"
PRINT string$
LET string$(9:18)= "Beihai"
LET string$(16:25)= "Baita"
PRINT string$
LET string$(9:20)=" "
PRINT string$
LET string$(Maxnum:0)= "→shanghai"
PRINT string$
END
OK.RUN
Tian Anmen
Beijing Tian Anmen
Beijing Tian Anmen Guangchang
Beijing Beihai Baita
Beijing
Beijing→shanghai
```

第9-2　字符串函数的应用

在第七章的标准函数中，曾介绍过用于字符串处理的字符串函数，每个特定的字符串函数可完成特定的字符串操作，现将它们列出如下：

LEN (A$)	求A$串的长度。
POS (A$, B$, n)	由第n个字符起，查找B$串在A$中的位置。
TRIM$ (A$) ,	删除字串A$首尾部的空格。
STR$ (x)	将x数字变为相对应的字符串。
VAL (x$)	将用字符串表示的数字变为数字。
CHR$ (x)	将数字x变为相对应的字符。
ORD (x$)	将字符x$变为其相应的ASCII值。

下面将以实例说明这些字符串函数的应用。

一、LEN (A$) 函数的应用

LEN (A$) 用于测定一个给定字符串中包含的字符个数，其中A$是给定字符串，函数取值等于A$中的字符个数。

```
例如  LET A$= "good morning"
      LET x=LEN (A$)
```

x的值为12。

例5 设计一个程序，由输入的几个字符串中找出最长串。

```
! find the longest string
DIM x$(20)
INPUT PROMPT "How many strings? " : n
PRINT "Enter your strings; "
FOR i= 1 TO n
   INPUT x$(i)
NEXT i
LET long=len(x$(1))
LET place=1
FOR i=2 TO n
   LET ln=len(x$(i))
   IF ln>long THEN
      LET long=ln
      LET place=i
   END IF
NEXT i
PRINT "The longest string is"; x$(place)
PRINT "Number of characters: " ; long
END
```

二、子串位置函数POS (A$, x$, n) 的应用

函数中自变量有三个：A$和x$均是字符串，n是数值，表示A$中的起始查找位置。pos函数求在A$中第n个字符起，x$子串第一次出现的位置，所以函数值是一个数字。若从第n位起，A$中不存在x$这个子串，则函数值为0。若n值为1，则可以省略不写。

如T$= "true BASIC"

则Pos (T$, "ru", 1) 值为2，pos (T$, "IC", 4) 值为9。pos (T$, 'AB') 值为0，因为不存在子串"AB"。

Pos (A$, x$, n) 函数常用于字串查找，下面通过几个具体例子，说明应用。

例6 编写一个程序，由键盘输入一句英文句子，字词间用空格分开，程序可自动将句中的每个英文单词分开放入数组word$中，并将每一字词分行显示。

```
DIM word$(10)
LINE INPUT prompt "Enter the sentence" : text$
LET n=0
DO WHILE len(text$)>0
   LET n=n+1
   LET text$=trim$(text$)
   LET x=pos(text$, " " )
   IF x<>0 THEN
      LET word$(n)=text$(1:x-1)
      LET text$=text$(x:maxnum)
   ELSE
      LET word$(n)=text$(x:maxnum)
      LET text$=""
   END IF
   PRINT word$(n)
LOOP
PRINT
END
OK. RUN
Enter the sentence I am a student
I
am
a
student
```

例7 写一个用于文本编辑的子程序，其功能是用于修改一行文字，可用新的内容替代旧的。

要修改的一行文字可看作是一个字符串变量，在主程序中给定，作为实参传递到子程序。要改的字词作为子串Q$，先要到指定行中查找它的位置，若存在则按此位置以新的字词P$替代之；若不存在，则给出回答信息。

```
!  text editor
SUB change(t$)
   DIM word$(20)
   DO WHILE len(t$)<>0
      LET k=k+1
      LET t$=trim$(t$)
      LET p=pos(t$, " ")
      IF p<>0 THEN
         LET word$(k)=t$[1:p-1]
         LET t$=t$[p:maxnum]
      ELSE
         LET word$(k)=t$[p:maxnum]
         LET t$=""
```

```
        END IF
  LOOP
  FOR i=1 TO k
      IF word$(i)=q$ THEN
        LET t$=t$&" "&p$
        LET find$="true"
    ELSE
        LET t$=t$&" "&word$(i)
    END IF
  NEXT i
  LET t$=trim$(t$)
  IF find$<>"true" THEN PRINT q$; "not found! "
  END SUB
  LINE INPUT prompt "Enter the sentence: ": text$
  INPUT prompt "Enter the old string: ": q$
  INPUT prompt "Enter the new string: ": p$
  CALL change(text$)
  PRINT text$
  END
```

例8 编一个按姓名或地址查找有关人员的程序，在主程序中建立三个字符串数组：name$ (20) ，address$ (20) ，tele$ (20) ，三个数组中下标相同的元素，是同一个人的姓名、地址和电话号码。子程序中输入姓名或地址。按姓名查找时，输出所有同地址人的信息；按地址查找时，则输出所有同地址人的信息。查不到时，返回查无此人的信息。

```
DIM name$ (20), add$ (20), tele$ (20)
SET MODE "HIRES"
LET n=0
DO WHILE MORE DATA
   LET n=n+1
   READ name$(n), add$(n), tele$(n)
LOOP
LINE INPUT prompt "输入查找姓名或地址: ": a$
FOR i=1 TO n
   IF pos (name$ (i) , a$) <>0 OR pos (add$ (i) , a$) <>0 THEN
      LET find$="true"
      PRINT name$ (i) , add$ (i) , tele$ (i)
   END IF
NEXT i
IF find$<>"true" THEN
   PRINT"查无此人! "
END IF
DATA 张民, 合肥寿春路15号, 243002
DATA 李奇, 合肥长江路215号, 240009
DATA 王志新, 合肥劳动路112号, 483221
END
```

三、字符串变换函数的应用

这类函数完成数字与字符串之间的类型变换。在程序设计中，有时希望将一个数字作为字符串来处理，以便于控制输出格式；有时又需将以字符串形式表示的数字串变成数字进行计算处理；有时还会出现对字符与其ASCII码之间的变换要求。凡此种种情况，均需要用到类型转换函数。

变换函数有两组：str$(x) ，val(x$) 与chr$(x) ，ord(x$) 。

1.STR$ (x) 与val (x$)

STP$(x)函数将自变量x的数字转换为字符串，所以x是数值型变量。val(x$) 将字符串x$转换为数字，但x$必须是数字组成的字符串，其中第一个字符可以是正负号，数字中间可以包括一个小数点。

如 LET x=577.58
LET S$=str$(x)

则 S$的值是“577.58”

又如LET x$=“—118.58”
LEL V=lva(x$)

则V的值是-118.58。

2.CHR$(x) 与ord(x$)

CHR$ (x) 函数的自变量是数值，函数值是以自变量的数值为ASCII码的字符。ORD (x$) 的自变量是一个字符，函数值是这个字符的ASCII码。

如 PRINT chr$(65) ；chr$(66)

将输出AB，因为A，B的ASCII码是65和66。

又如 PRINT ord (“A”)，ord (“B”)

将输出65　　66。

例9 对给定的任一不超过8位数的正整数，要求在其最高位前以0补足到8位，然后输出。在对一事物的编码中，通常采用这种固定位数方法处理，如自行车牌号，人员编码等。但对数值来说，有效数字前的0是不能输出的，所以要将数值改用字符串形式输出。程序为：

```
LET A$="0"
INPUT n
LET x$=STR$(n)
LET m=LEN(x$)
IF m<8 THEN
   FOR i=1 TO 8-m
      LET a$=a$&"0"
   NEXT i
LET x$=a$&x$
END IF
PRINT x$
END
```

例10 编一个简单的密码生成程序。要求输入的若干个字符（不超过20个，且只包括A～Z26个字母和空格，没有其它符号）各自换成其后的一个字母。即A换为B，B换为C，……Z换为A。空格保持不变。

```
LET text1$=“ ”
FOR i=1 TO 20
LET text1$=text1 $& “ ”
NEXT i
INPUT text$
FOR i=1 TO 20
IF text$(i:i)<>“ ” THEN
    IF text$(i:i)=“Z” OR text$(i:i)=“z” THEN
        LET n=ORD (text$ (i:i))
        LET n=n-25
        LET text1$ (i:i)=CHR$ (n)
    ELSE
        LET x$=text$(i:i)
        LET n=ORD(x$)
        LET n=n+1
        LET text1$(i:i)=CHR$(n)
    END IF
END IF
NEXT i
PRINT text1$
END
```

习 题 九

1. 下列程序输出是什么？

```
LET A=42
LET B=36
LET C= (42+36) /100
LET A$=STR$ (C) &“%”
PRINT “The result is” ; A$
END
```

2. 在下列○处填入合适的关系运算符，使关系表达式的值为真。

“BASIC” ○ “PASCAL”
“Shanghai” ○ “Beijing”
“man” ○ “men”

3. 设计一个程序，打印出由字母组成的13层的金字塔，形如

```
    A
  A B C
A B C D E
    ·
    ·
    ·
```

4. 编写一个将若干个国名按字典顺序排序的程序。

5. 编写一个程序，将任意输入的若干个英文字母按与输入时相反的顺序输出。

6. 设计一个程序，用READ/DATA语句读入数据。再按姓名查找某学生的数据。数据包括：姓名，性别，班级，年龄，籍贯。

7. 修改第6题的程序，改为按年龄段（上下限）查找，输出有关数据。

8. 编辑一个句子，当句子中连续出现多个空格时，只保留一个，删去多余的空格。写成一个子程序。

9. 一个句子由多个单词组成，其间由空格隔开。试编程序统计句子中含有一个、二个、三个、……、十个字母的词各有多少个。

第十章　格式化输入与输出

前面我们在输入数据与输出数据时所用的INPUT语句和PRINT语句，都是按这两个语句固有的格式输入和输出。虽然可以通过所采用的不同分隔符号（如分号，逗号或语句末尾不加任何符号）以及PRINT语句配合TAB函数等方法来获取输入、输出的不同格式，但在实际中，有时还不能满足用户需要。因此，本章将通过光标定位控制，格式化输出等语句使输入与输出的数据格式能够由用户自行选择，以满足各种不同的需要。

§10-1　屏幕的光标定位控制

微型计算机系统将显示器的屏幕按坐标法，纵向分为25行（Line），横向分为80列（Column）称为字符显示模式。所谓光标，是用以表示当前屏幕显示位置的标记，光标所在位置就是字符显示的起始位置，随着字符的输出，光标位置作对应的变动。在屏幕上光标是一个“一”形亮线。

下面的程序包括有输入输出语句，我们可以通过这个程序，说明光标位置变化的情况。

例1

```
CLEAR
PRINT "How are you? "
LINE INPUT a$
PRINT a$
PAUSE 2
END
```

执行第一条CLEAR语句后，清除屏幕上的当前内容，光标停留在第一行第一列。这就是后面输出语句所要显示的起始位置。

执行第二条语句时，光标由第一行第一列位置开始显示How are you? 等12个字符，光标逐列由左向右移动，因为PRINT语句结尾无分号，也无逗号，所以显示了字符“？”之后，光标移向下一行（第二行）的第一列。

执行第三行时，就在第二行第一列显示提示输入的问号“？”，光标移后两格等待显示键盘输入的字符串。每输入一字符，光标自动向右移动一格，直到键入回车（ENTER键）后，光标移向第三行首列。

执行第四条语句时，从光标所在位置开始显示键入的内容。

暂停2秒之后，清屏幕恢复到运行前的分窗显示状态。

一、光标定位语句（SET CURSOR 语句）

为了能够在屏幕上任一指定的位置显示要显示的内容，必须予先将光标设置在要显示的起始位置。True BASIC提供了光标定位语句，该语句一般格式为：

SET CURSOR　行，列

其中，行与列是数字表达式，其取值范围分别限制为1～25和1～80。

二、格式化输入的控制

有了上述光标定位语句，我们就能够控制输入及输出的显示位置。

例2

```
SET MODE "hires"
PRINT "How are you? "
SET CURSOR 4, 25
LINE INPUT a$
SET CURSOR 5, 25
PRINT a$
PAUSE 2
END
```

程序中第一个输出语句的内容，其显示的格式没有加以控制，按PRINT语句固有的格式，从第一行第一列开始显示，之后，光标移到第二行第一列。由于第三行设置了光标的位置（移到4行25列），从而使以后的显示位置，起始于4行25列。输入数据结束，按回车键（Enter）光标自动移向5行1列。同理，第二个set cursor语句，又使下一个Print语句的数据显示起始位置从5行1列改变到5行25列。

加了光标位置定位语句，轻易地改变了输入数据的显示位置。为了进一步控制输入的格式，还可以利用GET KEY语句，对输入的数据的长度加以格式化。

GET KEY var语句的功能是：执行该语句时，程序指定运行，等待用户键入一个字符后，将该字符键入的ASCII码值赋给变量var，然后继续运行。利用此功能，就能控制输入数据的格式化。

例3

```
SET MODE "80"
INPUT PROMPT "How many characters want to input? " : n
INPUT PROMPT "The cursor move in line:  (1--20) " : line
INPUT PROMPT "The cursor move in column (1---50) " : column
SET CURSOR line, column
PRINT repeat$ (chr$ (219) , n)
FOR  i=0  TO  n-1
      GET KEY v
      SET CURSOR line, column+i
      PRINT chr$(v)
      LET a$=a$ & chr$ (v)
NEXT i
PRINT
PRINT "characters entered: " ;
PRINT a$
PAUSE 5
END
```

本程序前三行的数据输入格式，是不加以控制的，也就是按INPUT语句固有的格式，用户逐行输入字符长度、光标的位置（行值和列值）。经第一个设置光标语句，将光标移到所给定的位置，并在此位置开始显示一条光带，它的长度就是前面输入的字符长度。程序中加入循环结构，目的是等待一一输入要输入的字符，并在上述的光带上显示相应的字符（因为循环体中的SET CURSOR语句，将光标重新移动光带的相应位置，并且每输入一个字符，光标定位一次），待输完规定长度的字符串，自动结束输入。程序最后重新显示所输入的全部数据。

例4 显示一个菜单程序。

```
CLEAR
    SET CURSOR 2,10
    PRINT repeat$ ( "*" , 60)
    SET CURSOR 3, 30
    PRINT "1. Sum and mean number"
    SET CURSOR 4,30
    PRINT "2. Guess"
    SET CURSOR 5,30
    PRINT "3, Exit"
    SET CURSOR 6,10
    PRINT repeat$ ( "*" ,60)
    SET CURSOR 7,25
    PRINT "Enter your selection"
END
```

三、光标位置查询语句（ASK CURSOR）

为了了解当前光标所在位置，求出其行与列的当前值，True BASIC提供了一条询问光标位置语句，该语句的一般格式为：

ASK CURSOR Row, COLUMN

执行此语句以后，row和column两个变量的值就是光标所在的行与列的当前值。

此外，ASK MAX CURSOR Height, Width语句将会提供所用屏幕窗口的最大行坐标值，（高度）和最大列坐标（宽度），并把两个值分别赋给予变量Height和Width。

例5

```
SET CURSOR 2,5
ASK CURSOR x,y
PRINT x,y
SET CURSOR 10,5
ASK CURSOR X,Y
PRINT x,y
ASK MAX CURSOR h,w
SET CURSOR 25,5
PRINT h,w
PAUSE 5
END
```

四、关闭光标与恢复光标

一般情况下，光标设置在“开”状态，所以屏幕上有“-”形符号标志光标所在位置。有时要求关闭光标，此时，可用语句：

```
SET CURSOR "OFF"
```

执行上述语句后，光标就从屏幕上消失了。如果要重新显示光标，可用语句：

```
SET CURSOR "ON"
```

语句ASK CURSOR var$执行之后，可以从变量Var$中取得“ON”或“OFF”，了解当前光标所处的状态开启或关闭。

§10-2　自选打印格式语句

一、自选打印格式语句（PRINT USING语句）

一般格式为：

```
PRINT USING"<格式字串>"：<表达式表>[，][；]
```

其中，格式字串是字符串表达式，用来指定表达式表（或自然输出项）显示格式。如果在语句未尾有可选项[，]或[；]，则输出显示后光标停在输出项的结尾处，否则光标就移向下一行开始位置。

格式字串除了包括规定的格式代码之外，还可以包括任何字符。格式代码只是用来控制表达式表输出的格式，符号本身并不输出，而格式代码以外的字符将与输出项一起显示输出。格式用的代码有：

<>+-$*%#，.^等11个符号。

二、数字的格式符号

在使用PRINT USING语句来控制数字的格式化输出时，格式代码可用以下三种符号。

#——每个#号代表一位数字，数字前置的零由空格符来显示，如：“####”格式代码显示56时，其结果为：“ 56”。

%——每个%号代表一位数字，数字前置的零由0来显示，如“%%%%”格式代码显示56时，其结果为：“0056”。

*——每个*号代表一位数字，数字前置的零由*号来显示。如：“****”格式代码显示56时，其结果为：“**56”。

在格式符号的首部可加以下符号。

+——数字前面可显示“+”或“-”。

-——数字前面可显示“-”，若是正数则不显示。

$——数字前面显示$。

此外，格式代码中间还可以插入以下各符号。

，——将逗号插入数字中适当的位置。

.——小数点定位显示。

^——数字科学表示法中显示指数部分。

在格式代码中，至少有一个数字格式字符，用它来描述输出数字的格式。

例如： PRINT USING "####"：5

表示数字前置的零以三个空格符号显示： 5。

而PRINT USING "%%%%"：5则以0来表示三个前置的零，显示为"0005"格式。

注意的是不能把不同的格式符混合在一起构成的格式项来描述输出格式。如PRINT USING "%%##"：5是不合法的。

为了说明在十一种格式代码中，能对数值格式控制的九种格式代码的作用，我们通过一些实例逐一介绍。

例6 运行下面程序观察其结果。

```
DO WHILE MORE DATA
   READ a,b,c
   PRINT USING "###" : a,
   PRINT USING "%%%" : b,
   PRINT USING "***" : c
LOOP
DATA 0, 0, 0, 1, 1, 1, —1, —1, —1, 12, 12, 12, 12.26, 12.26, 12.26
DATA—12.56, —12.56, —12.56, 123.4, 123.4, 123.4, —123.4, —123.4, —123.4
END
OK.RUN
        0              0  0  0          *  *  0
        1              0  0  1          *  *  1
     —  1              —  0  1          —  *  1
     1  2              0  1  2          *  1  2
     1  2              0  1  2          *  1  2
  —  1  3              —  1  3          —  1  3
  1  2  3              1  2  3          1  2  3
  *  *  *              *  *  *          *  *  *
```

由例6可以把#、%、*的功能总结为：

(1) 每个"#"号代表一个数值输出位。

(2) 数值右对齐。多余"#"号以空格补齐。

(3) 小数部分不显示，并有自动四舍五入的功能。

(4) 负号在最左侧，正号不显示。

(5) 数值超过定义的格式长度而不够输出时，则显示出与"#"号个数相同的"*"号，表示溢出。

(6) "%"号的功能与"#"号基本相同，区别仅在于多余的"%"号以零补齐。

(7) "*"号的功能与"#"号基本相同，区别仅在于多余的"*"号以"*"补齐。

例7 运行下面程序分析输出结果。

```
DO WHILE MORE DATA
   READ a, b, c,
   PRINT USING "###.##" , a,
   PRINT USING "%%%.%%" , b,
    PRINT USING "***.**" : c
LOOP
DATA 0, 0, 0, 1, 1, 1, —1, —1, —1, 12, 12, 12, 12.26, 12.26, 12.26
DATA—12.56, —12.56, —12.56, 123.4, 123.4, 123.4, —123.4, —123.4, —123.4
END
OK.RUN
      . 0 0          0 0 0 . 0 0          * * * . 0 0
    1 . 0 0          0 0 1 . 0 0          * * 1 . 0 0
  — 1 . 0 0          — 0 1 . 0 0          — * 1 . 0 0
  1 2 . 0 0          0 1 2 . 0 0          * 1 2 . 0 0
  1 2 . 2 6          0 1 2 . 2 6          * 1 2 . 2 6
— 1 2 . 5 6          — 1 2 . 5 6          — 1 2 . 5 6
1 2 3 . 4 0          1 2 3 . 4 0          1 2 3 . 4 0
* * * * * *          * * * * * *          * * * * * *
```

由例7可以看到，格式字串的“.”号起小数点定位输出作用，并总结为：

(1) 数值按小数点对齐输出。

(2) 整数部分多余的“#”号以空格补齐；多余的“%”号以零补齐；多余的“*”号以“*”号补齐。

(3) 小数部分有四舍五入的功能。

(4) 小数部分多余的“#”号或“%”或“*”均以零补齐。

(5) 数值超过格式字串定义的长度而不够输出时，均以全“*”号输出，以表示溢出。

格式字串中，常在适当位置输入“，”号，起着分位点作用。若在“，”号左边也没有数字输出，则“，”号以空格补位输出。

例8 运行下列程序观其结果。

```
PRINT USING "##, ###" : 56
PRINT USING "##, ###" : 2345
PRINT USING "##, ###" : 17950
PRINT USING "##, ###" : 375
PRINT USING "##, ###" : 136724
PRINT USING "##, ###, ###.##" : —2.5E6
END

OK.RUN
        5 6
  2 , 3 4 5
1 7 , 9 5 0
      3 7 5
* * * * * *
—2, 500, 000.00
```

在格式字串中的“^”号的作用是使数值按科学表示法（或指数形表示法）输出。

例9 运行下面程序，分析执行结果。

```
PRINT USING "##^^^" : 2.5E6
PRINT USING "#.#^^^": 2.5E5
PRINT USING "##.#^^^": 2.4E5
PRINT USING "##.###^^^": —185E6
PRINT USING "##.###^^^^": —185E6
PRINT USING "##.###^^^^^": —185E6
PRINT USING "##.###^^^": —185E—5
PRINT USING "##.###^^^": —185E—12
END

OK.RUN
25e+5
2.5e+5
24.0e+4
—1.850e+8
—1.850e+08
—1.850e+008
—1.850e—3
*********
```

对于“ ^”号的功能总结为：

(1) “ ^”号必须放在格式字串中的尾部。

(2) “ ^”号至少要有三个，否则出错。而“ ^”号个数减2是指数数字部分所占位数。

(3) 指数部分的输出值由计算机自动调整。

在格式字串中，我们把+、—、$符号称为首部代码。因为它们在数字代码（#、%、*）之前。

“+”号作用：在数字前面显示出数值的正负号。

“—”号作用：在数字前面只显示数值负号，正号用空格代替。

“$”号作用：在数字前面显示出美元符号“$”。

若希望正负号或$符紧挨着数字显示，则用连续多个相同的首部代码。

例10 分析下面程序的运行结果。

```
PRINT USING "+###.##": 24.2
PRINT USING "+###.##": —24.2
PRINT USING "+###.##": 165.23
PRINT USING "+###.##": —472.51
PRINT USING "+###.##": 12345.2
PRINT USING "—###.##": 412.34
PRINT USING "—%%%.%%": 54.2
PRINT USING "—%%%.%%": —54.23
PRINT USING "$###.##": 12.5
PRINT USING "$###.##": —24.34
PRINT USING "$***.**": 45.6
PRINT USING "$***.**": —37.6
PRINT USING "+++##.#": 12.56
PRINT USING "+++##.#": —12.56
PRINT USING “++++.#” : —543.2
PRINT USING “++++.#” : 1543.2
PRINT USING “——%%.%” : 12.5
PRINT USING “———%.%” : —12.5
PRINT USING “$$**.*” : —12.5
PRINT USING “$$$*.*” :—27.88
PRINT USING “$$$$.#” :125.7
PRINT USING “$$$$.#” :3754.92
END

OK.RUN
+24.20
—24.20
+165.23
—472.51
****
412.34
054.20
—054.20
$12.50
$—24.34
$*45.60
$—37.60
   +12.6
   —12.6
  —543.2
 ****
    12.5
   —12.5
$—12.5
 $—27.9
  $125.7
  ****
```

三、字符串的格式化输出

字符串可以任意格式代码定义输出格式。当字符串小于格式字符串所规定的长度时，则显示的字符串显示在中间位置，两边由空格充填，如果加有首部字符“<”或“>”时则显示结果将靠左对齐或靠右对齐。同数字显示一样，字符串长度大于规定的空间时，显示结果均为＊。

为了说明这些格式代码对字符串格式的控制，请看下面实例。

例11 分析下面程序的运行结果。

```
PRINT USING “*****” :  “FIELD”
PRINT USING “**.**” :  “LOOP”
PRINT USING “$*—##” :  “ABC”
PRINT USING “—*.**” :  “OK”
PRINT USING “+%^*^” :  “T”
PRINT USING "#.*%$" :  "BEIJING"
PRINT USING ">####3" :  "FIELD"
PRINT USING "<##.#" :  "BOOK"
PRIPT USING ">$%*, " :  "but"
PRINT USING "<%.**" :  "OK"
PRINT USING ">#*^" :  "SHANGHAI"
END

OK.RUN
FIELD
LOOP
 ABC
 OK
  T
 *****
FIELD3
BOOK
  but
OK
 ****
```

习 题 十

1. 以下字符哪些可作为格式化输出的格式符号？

(a) #　(b) %　(c) $　(d) ?　(e) <　(f) ;

(g) +　(i) =　(j) *　(k) >

2. 判断下述用于数字显示的格式字符串是否合法？

(a) "$$###"　(b) "%%###"　(c) "+##.##"

(d) "—##.##"　(e) ">##"

3. 写出下述语句执行后的结果。

```
PRINT USING "$$$$$$.##" : A
```

A的取值分别为23.45, 2468, 25, 345, 23456.5

4. 写出下述程序执行后的结果。

```
LET a$= "ABC"
PRINT USING "#####" : A$
PRINT USING "<#####" : A$
PRINT USING ">#####" : A$
PRINT USING "##" : A$
```

第十一章　文　件

§11-1　文件的基本概念

前面所学程序中的数据通常以DATA语句附于程序中，这种方式在处理大量数据时，是很不方便的。一方面数据要占程序中相当大的篇幅和内存，维护修改也不方便；此外，程序不与数据分开也限制了程序的通用性。所以当代计算机应用中通常将程序和数据分开，数据被单独地存入文件。这样做的优点是：数据与程序分开，增加了程序设计的灵活性与通用性：一个程序可对不同数据文件进行处理获得不同的结果；独立的数据文件便于数据的保存、维护与修改；数据文件因存于“外存”——磁盘上，容量大，可以不受内存空间限制；磁盘文件可以永久保存。这种用于存储信息数据的文件称为数据文件。

一、数据和文件

1.信息

所谓信息是指通过各种方式传播的，可被感受的声音、文字、图象等所表征的某一特定事物的消息，情报或知识。换句话说，信息是对客观事物反映，是为某一特定目的而提供的决策依据。当然，在处理问题时，人们只关心对其有用的信息。

2.数据：

所谓数据是指表达信息的某种符号，是信息的一种量化表示，是信息的载体。数据反映信息，而信息依靠数据表达。

表达信息的符号必须遵从一定的约定，根据约定，现实中的一切信息都可以表示成数据并能被计算机加以储存和处理。

计算机只能存储数据，因此，必须人为地把信息转换成可以被计算机接受的数据，但不管什么数据，都是以二进制形式存储在计算机内并被计算机加工处理的。

3.文件

所谓文件就是数据的集合，对计算机来说，文件是指存储在磁盘上的数据集合。它们在程序运行之后继续存在，并可以被以后的程序所使用。True BASIC中，文件一共有三种：①正文文件（顺序文件）；②记录文件；③字节文件。

二、文件名

为了标识一个文件，每个文件都应该有它的文件名。同一个介质上不能有两个相同的文件名。True BASIC规定文件名的格式为：

[〈盘符〉:]〈文件名〉[.〈扩展名〉]

盘符：即磁盘所在的驱动器号，通常规定A代表第一台磁盘驱动器，B代表第二台驱动器。盘符后一定要加冒号。当盘符和冒号缺省时，隐含指当前正在工作的“当前驱动器。”

文件名：实际上是文件名的主干，一般用1～8个字符组成，多数情况下，由字母和数字

组成，如BU08，A-1，ABC等都是合法的文件名主干。当然，也可以应用一些专用的字符，如$$$$，aaa等，也是合法的，但要注意有些专用字符是不可用的，如空格和逗号等，这里我们提倡用字母和数字，且长度是1～8个。

DOS系统还有一些专用的设备文件名，在给一般文件起名时，决不要使用这些保留设备文件名，以免出错。这些保留设备文件名有：

CON：　键盘或屏幕；
AUX：　第一台异步通信适配器接口；
COM1：　第一台异步通信适配器接口；
COM2：　第二台异步通信适配器接口；
PRN：　第一台并行打印机接口；
LPT1：　第一台并行打印机接口；
LPT2：　第二台并行打印机接口；
LPT3：　第三台并行打印机接口；
NUL：　空设备。

扩展名：其长度不超过三个字符，用来表示文件的属性，它必须用“•”号与文件名分隔。如：

•TRU　表示True BASIC源程序；
•FOR　表示FORTRAN源程序；
•COM　表示可执行的二进制代码文件；
•SYS　表示系统文件；
•LIB　表示库文件；
•DAT　表示数据文件。

扩展名所用字符的规定与文件名主干一样，但它也可以省略不写。

三、通道（又称通道号）

要对磁盘上的一个数据文件进行读写操作，必须首先开辟一个内存缓冲区，以建立必要的通道。而文件号就是规定了对这个文件进行操作的一个通道，所以也称通道号。当由OPEN语句打开一个通道时，这个通道就与你要使用的某个文件就建立了联系。当由CLOSE语句关闭某个通道时，这种联系就结束了。当程序结束时这种联系也被关闭。

通道定义为：

#n

其中“#”号为通道识别符；n为文件的通道号，它可以是数字常数、数字变量或数字表达式。而数字表达式的值必须是0～999的正整数，若结果是小数，机器自动四舍五入。

通道#0是专用的，它特指屏幕和键盘。此通道总是打开的，不能关闭。

§11-2　顺序文件（文本文件）

一、顺序文件的特点与操作步骤

1.顺序文件中的数据结构

(1)顺序文件中的数据是以可显字符形式按行存放的。每行有一定的字符限制，称为Mar gin（区宽），不作声明时，默认为80列（与字符屏幕相同）。

(2)每行中字符数可以不同，最后是行结束标志。

(3)顺序文件中数据的读或写，只能由文件开始，一行行地顺序进行，不能由任意指定的地方读出或写入。

(4)顺序文件结束时有一个结束标志，通常是“∧Z”（或Cre+Z）。

2.顺序文件的操作步骤

顺序文件在使用时应遵循以下步骤：

(1)打开文件：分配文件通道号给要打开的文件名，并说明文件的访问方式，文件类型，建立模式。

(2)写入或读出数据：写入数据只能从文件开头或文件末尾处顺序写入，读出数据时只能从文件开头顺序读出。

(3)关闭文件：当文件操作结束后，要关闭打开的文件，重新访问该文件时再打开。

(4)顺序文件的写入使用PRINT#n:〈表达式〉或PRINT#n，USING A$:〈表达式〉语句；读出时使用INPUT#n：〈变量表〉或LINE INPUT#n：B$语句。

二、顺序文件的建立

1.顺序文件的建立——OPEN语句

文件操作的第一步是使用open语句打开要处理的文件，其格式为：

open#n：name〈文件名〉，access〈访问模式〉，create〈建立模式〉，organization〈类型〉，其中n 为通道号，为1～999之间的任意整数。〈文件名〉为要打开的文件名，与DOS系统中的命名规则相同，可用字符串常量也可用字符串变量表示。

〈访问模式〉规定打开文件的访问方式，可有以下三种选择：

(1)input—只能读出文件数据，

(2)output—只能向文件写入数据，

(3)outin—既可写入也可读出。省略access项说明时，隐含文件为outin。

〈建立模式〉规定文件建立的类型，可有以下三种选择：

(1)NEW—建立一个新文件，

(2)OLD—打开已经存在的文件，

(3)NEWOLD——建立新文件，若已存在，则打开老文件。如果省略create项说明时，默认为OLD方式。organization TEXT说明文件类型为正文文件（即顺序文件）。

例如：open#1：name“goze”，create new，organization text

此语句设置通道号为1，文件名为goze，建立模式为new，文件类型为text。执行此语句后，磁盘当前目录中则写入一个goze·tru的文件，内容为空（尚待写入）。

2.顺序文件内容的写入

建立顺序文件是很容易的。顺序文件建立时，需要向文件输入数据，而数据的写入是通过顺序文件的写语句（PRINT语句）来实现的。但应给出通道，以指明向哪个文件写入。

PRINT语句有以下几种格式：

(1)PRINT#n：〈表达式〉

(2)PRINT#n，USING格式字串：〈表达式〉

(3)MAT PRINT#n：〈数组1〉，〈数组2〉，…

(4)MAT PRINT#n，USING格式字串：〈数组1〉，〈数组2〉，…

3.关闭文件——CLOSE语句

一个文件建立或读写之后，若不使用，可使用CLOSE语句关闭，格式为：

CLOSE#n

其中，n为打开文件时使用的通道号。

例1 下面程序是建立一个顺序文件“goze”，该文件含有姓名、地址和工资三项内容。

```
!open text file, and print goze
!
OPEN #1: name "goze" , organization text, create new
PRINT  #1
PRINT  #1:  "zhigong goze qingdan"
PRINT  #1
LET nm$= "<# # # # # # # # # # # # # # # # # # # # $ # # #.# #"
PRINT #1, using nm$:  "Wang ping Beijing" , 125.50
PRINT #1, using nm$:  "Lilin Shanghai" , 100.80
PRINT #1, using nm$:  "Ying lio Tianjin" , 205.00
PRINT #1, using nm$:  "zhang gong Nanjing" , 135.60
PRINT #1, using nm$:  "Zhao ming Wuhan" , 156.90
PRINT #1, using nm$:  "Jing nan Hangzho" , 86.40
PRINT #1, using nm$:  "Lio bang Xian" , 178.50
PRINT #1, using nm$:  "Ye qing Guangzho" , 315.80
CLOSE #1,
END
```

三、从顺序文件读取数据

从顺序文件读取数据是通过INPUT语句来实现的，INPUT语句必须给出通道，以指明是从哪个正文文件读取数据。

INPUT语句有以下几种格式：

(1)INPUT#n：〈变量表〉

(2)LINE INPUT#n：〈字符变量表〉

(3)MAT INPUT#n：〈数组1〉，〈数组2〉，…

(4)MAT INPUT#n：〈字符数组1〉，〈字符数组2〉，…

注意：顺序文件读语句（INPUT语句）不能使用PROMPT选择项，即不能有文字提示。

从顺序文件读取数据需要测试文件是否读完，即测试文件是否结束。True BASIG有下面两种测试方法：

(1)MORE#n

如果文件指针没见尾，其逻辑值为真。如果文件指针已指向文件的末尾，逻辑值为段。

(2)END#n

如果文件指针没见尾，其逻辑值为假，如果文件指针指向文件末尾，其逻辑值为真。

例2 从例1所建立的顺序文件“goze”中读取数据，并从打印机输出。

```
! open text file, and print goze
!
OPEN#2: printer
OPFN a$ (10000)
OPEN#1: name"goze" , organization text, access outin
LET i=0
DO WHILE MORE #1
   LET i=i+1
   LINE INPUT#1: a$ (i)
LOOP
PRINT  #2
FOR k=1 TO i
PRINT#2: a$ (k)
NEXT k
CLOSE  #1
END
OK.RUN
zhigong goze qinngdan
Wang ping Beijing         $125.50
Li lin Shanghai           $100.80
Ying lio Tianjin          $205.00
Zhang gong Nanjing        $135.60
Zhao ming Wuhan           $156.90
Jing nan Hangzho          $86.40
Lio bang Xian             $178.50
Ye qing Guangzho          $315.80
```

程序中首先用OPEN语句定义了通道#2为打印机，此时通道号#2就与打印机确定了联系。OPEN语句完成两个功能：第一将打印机打开；第二确定了通道#2与打印机联系。实际上，通道#2就是打印机的临时代号，此后可以反复使用语句PRINT#2。

用more#1控制DO循环。若文件结束，more#1为假，否则为真，这样可以一直读到文件末尾。

程序最后以CLOSE#1关闭文件，如果程序执行到END前无CLOSE语句，则文件在执行END语句时自动关闭。

四、顺序文件增加数据

在讨论顺序文件增加数之前，提请用户注意顺序文件写入与读出的限制。

1.顺序文件写入的限制

一个顺序文件打开后要写入数据时，第一个限制是必须从头或从尾写入。如果文件是一个老文件，打开后有一指针处于文件的开头，但这时不可写入，因为从头写入的另一条件是文件必须是空的，要从头写入一个老文件，必须用ERASE语句，将文件内容擦除。但文件仍然存在，只是空的。ERASE语句常紧跟在打开文件OPEN语句之后。

ERASE语句的一般格式：

ERASE#n

2.顺序文件读出的限制和写入的格式控制

由顺序文件中读出数据也是有限制的：第一个限制是必须从文件的开始顺序读出；第二个限制是一次必须读出一行，而不能一个个地读出变量。但由于按标准显示格式一行只能显示5个数据，因行宽80，数据间距16。为此引入控制文件写入时的格式，True BASIG提供了行宽及数据间距的控制语句，其一般格式为：

```
SET#n MARGIN〈数字表达式〉
SET#n ZONEWIDTH〈数字表达式〉
```

其中，n为打开文件的通道号，MARGIN为行宽说明，其后数学表达式的值应在80-32767之间；ZONEWIDTH为数据间距说明，其后的数值表达式的值应在1～Margin之间。

忽略MARGIN设置，默认为80列；忽略ZONEWIDTH默认为16。

3.移动文件指针语句

文件没有关闭前，要读出文件内容时，必须用SET POINTER语句将指针移至文件开头。下述语句是等效的：

```
SET #n: POINTER BEGIN
RESET #n: Begin
```

如果将指针设置到文件末尾，可用语句

```
SET POINTER #n: END
RESET #n: END
```

为了对顺序文件增加数据，必须把文件指针移到文件的末尾。如何把文件指针移到文件的末尾呢？应当通过下面语句

```
SET#1: POINTER END
```

来实现，这时才能增加数据。

例3 设“goze”文件中原有数据为：

```
zhigong goze qingdan
Wang ping Beijing    $125.50
Li lin Shanghai      $100.8
Ying lio Tianjin     $205.00
Zhang gong Nanjing   $135.60
Zhao ming Wuhan      $ 156.90
Jinng nan Hangzho    $86.40
Lio bang Xian         $178.50
Ye qing Guangzho     $315.80
OK
```

现在欲增加如下四条数据：

```
Ging hua Nanchang      235.00
Xu jianhua Cuzho       164.50
Dong zhiwen chengdu    185.40
Xu ming Hefei          115.80
```

数据增加之后，我们把顺序文件“goze”的全部内容由打印机输出，这时文件指针要移到文件开头。

程序及运行结果为：

```
! open text file, and add data
!
OPEN #2: printer
OPEN#1: name"goze" , organization text, access outin
LET nm$="<# # # # # # # # # # # ## # # # # # # # $ # # #.# #"
SET#1:POInfer END
DO
   INPUT xj$
   IF xj$="end" OR xj$="END" THEN EXIT DO
   INPUT gz
   PRINT #1, using nm$: xj$, gz
LOOP
SET #1: POINLE BEGIN
DO WHIE MORE #1
   LINE INPUT #1: line$
   PRINT#2: line $
LOOP
END
XK.RUN
zhigong goze qingdan
Wang ping Beijing     $125.50
Li lin Shanghai       $100.80
Ying lio Tianjin      $205.00
Zhang gong Nanjing    $135.60
Zhao ming Wuhan       $156.90
Jing nan Hangzho      $86.40
Lio bang Xian         $178.50
Ye qing Guangzho      $315.80
Ging hua Nanchang     $235.00
Xu jianhua Cuzho      $164.50
Dong zhiwen Chengdu $185.40
Xu ming Hefei         $115.80
```

五、顺序文件的应用

1.向打印机输出数据

在True BASIC中，打印机被定义为一个特殊顺序文件，文件名为PRINTER，当程序的执行结果要送打印机打印时，可将打印机作为一个顺序文件处理，向打印机输出数据即相当于向PRINTER又写入要输出的数据，操作步骤与向一个文件的写入操作相同。

例4 将一组职工数据送打印机打印。

```
SET MODE "hires"
OPEN #1: PR!NTER
LET b$="%%%% <# # ## # # # # # $ $ $. ##"
PRINT #1: "编号  姓名  工资"
DO WHIE MORE DATA
   READ num, name$, pay
   PRINT #1 ,USING b$: num, name$, pay
```

```
LOOP
CLOSE #1
DATA 105, 王新, 105.00
DATA 201, 张民, 131.00
DATA 302, 刘金, 140.00
END
```

在程序的OPEN语句中通道号后的name可省略，因为PRINTER是系统保留的打印机文件名，organization也可省略，因打印机是一个默认的顺序文件。

2.简单行数据文件

用顺序文件建立职工工资数据文件，并可对数据文件增添数据，设工资表数据包括姓名、基本工资、物价补贴、卫生费、房租、水电、托儿费。

例5 设计顺序文件程序。

```
! 建立工资顺序文件（写）
  SET MODE "egaHIRES"
  DIM A$ (0 to 7) , T$ (3) , T (3, 7)
  INPUT PROMPT"是否追加 (Y/N) " : B$
  IF B$="Y" OR B$="y" then CALL append
  !
  FOR j=0 TO7
     READ a$ (j)
  NEXT j
  FOR i=1 TO 2
     READ t$ (i)
     FOR j=1 TO 6
        READ t (i, j)
        LET t (i, 7) =t (i, 7) +t (i, j)
     NEXT j
     LET t (3, 7) =t (3, 7) + (i, 7)
  NEXT i
  CALL write (a$, t, t$)
  DATA 姓  名, 基本工资, 物价补贴, 卫生费, 房租, 水电, 托儿费, 实发工资
  DATA 李  立, 56, 7.5, 1, -3, -7.5, -10.2
  DATA 王  为, 78, 7.5, 1, -4, -6, -9.3
  END
 ! SUB-1
 SUB write (a$ ( ) , t (, ) , t$ () )
    OPEN #1: name "gz-1", organization text, access output, create newold
    ERASE #1
    FOR i=0 TO 7
       PRINT #1: a$ (i)
    NEXT i
    FOR i=1 TO 2
       PRINT #1: t$ (i)
       FOR j=1 TO 7
```

```
            PRINT #1: T (i, j)
         NEXT j
      NEXT i
      CLOSE #1
   END SUB
   ! sub-2
   SUB append
      OPEN #1: name "gz-1", organization text, access output
      SET #1: PRINTER END
      DO
         INPUT PROMPT"姓名（输入 END 则追加结束）": xm$
         IF xm$="END" or xm$="end" THEN STOP
         INPUT PROMPT"基本工资，物价补贴，卫生费": A, B, C
         INPUT PROMPT"房租，水电，托儿费（应加负号）": D, E, F
         LET S=A+B+C+D+E+F
         PRINT #1: xM$
         PRINT #1: A
         PRINT #1: B
         PRINT #1: C
         PRINT #1: D
         PRINT #1: E
         PRINT #1: F
         PRINT #1: S
      LOOP
   END SUB
```

本程序由主程序及两个子程序组成，主程序根据是否已建立过工资文件，决定选择建立方式或增添方式，在建立方式中由Data语句提供数据（程序中只有两个人的数据，用户可以根据需要添加），读入数组并求得实发工资后调用于程序1（write）写入工资文件“GZ-1”。注意：这里定义的子程序均为外部子程序，所以在调用时要使用数组参数传递方式，子程序定义中使用了数组形式参数；调用时子程序名后括号中仅需写入数组名，注意实参与形参的一一对应。

在增添方式中调用子程序2，用户可由键盘输入数据，每输入一个人的各项数据（根据提示顺序），程序根据每送入一人后的对话确定是否结束输入。

注意，两个子程序中均使用了通道号参数#1，以将主程序打开的文件通道传送给外部子程序。

例6 由上例中建立的工资文件中读出数据并打印显示工资表。

```
! 读工资顺序文件…………打印表格
SET MODE "hires"
DIM t$ (10) , t (10, 7) , a$ (8)
OPEN 1: name "gz-1", organization text, access input
FOR i=1 to 8
   INPUT #1: a$ (i)
NEXT i
LET i=0
```

```
DO WHILE MORE #1
   LET i=i+1
   INPUT #1: t$ (i)
   FOR j=1 TO 7
      INPUT #1: t (i, j)
   NEXT j
 LOOP
!
FOR k=1 TO 7
   FOR j=1 TO i
      LET t (i+1, k) =t (i+1, k) +t (j, k)
   NEXT j
NEFT k
!  print the tab
PRINT tab (36) : "工资表"
PRINT tab (32) : "**************"
FOR j=1 TO 8
   PRINT USING "##########": a$ (j) ;
NEXT j
PRINT rcpeat$ ("-", 80)
FOR j=1 TO i+1
   IF j=i+1 THEN PRINT USIMG "###########": "合计";
   IF j<i+1  THEN PRINT USIMG "###########": t$ (j)
   FOR k=1 TO 7
      PRINT USIMG"- - - ### ##": t (j, k)
   NEXT k
   IF j=i THEM PRINT repeat $ ("-", 80)
NEXT j
END
```

本程序首先用open语句打开已建立的顺序式工资文件“GZ-1”，由于只从文件中读出数据，使用了access input选择。文件打开后，先将表头数据读出并放在A $数组中，然后分别读出每个人的姓名及工资收入与支出数据、实发工资等，并分别放在数组T $及T 中。为统计出每人的收支合计及实发工资数，使用了FOR循环，这里循环控制变量K的初值用的是2，这是因为第一列是姓名，不用合计。被合计的数据也放在数组T中，因为各项合计的结果是在最后一行打印出来，所以合计结果放入第n+1行中（即T数组第一维的下标为n+1，因为，共读入了n个人的数据）。

需要指出的是：在本例中写入文件时，每个数据独立写入一行，所以读出时也可以一个一个地读出数据，而且写入时是数字型变量，读出时仍读到一个数字型变量中，这里由系统自动地实现字符串到数字的变换；写入时是字符型变量，读出时仍是字符串。这样组织数据文件的优点是可以省略读出时的数据变换，缺点是一个人的数据被写入几行中，如不注意严格按写入时的数据顺序及数据类型读出时，很容易产生错误与混乱。

§11-3 记录文件

顺序文件由其组织形式决定，在写入或读出数据时，只能由文件开头顺序地逐步进行，既使要查询文件中的某一项内容，也必须将文件内容从头到尾地全部读出，这样就降低了速度，不便于记录插入、删除、更新等操作。故顺序文件多用于成批的数据保存，文本信息的记录。为了克服顺序文件的缺点，在信息处理与数据库应用中，常采用另一种文件组织方式——记录文件。记录文件是按记录组织存放数据的，文件中每一记录作为一个存储管理单位，具有相同的存储空间（记录长度）。在一个记录中，既可以存放字符串数据，也可以存放数字型数据，也可以混合存放，但其记录的总长度，不得超过预先规定的记预录长度。记录文件中的每一记录都隐含地具有一个记录号，指定记录号就可以对该记录进行读、写处理，因而可以随机地访问文件中的任一记录，故记录文件又叫随机文件。本节将介绍记录文件的建立、读写的管理方法。

一、记录文件的组织格式

1.记录的概念

在信息处理中经常与数据打交道，所谓数据是一种客观事物的数据化描述，例如，一个单位的工资管理信息，由每个人的有关数据项组成，这些数据项可包括：职工编号、姓名、基本工资、补贴、卫生费、房租、水电费、托儿费、实发工资等。每个人的数据项总和构成一个人的记录，单位职工数据库就是单位所有职工记录的集合。

2.记录文件中的记录组织格式

在True BASIC中提供的记录文件中，以记录为单位存取数据，每一记录具有相同的长度称为记录长度（recsize）；每一记录隐含一记录号，由1开始按自然计数对所有记录进行编号，指定记录号即可对该记录进行存取操作；文件打开后，隐含一指针，可以移动指针到任何指定的记录，指针指向的记录称为当前记录，任何时候只可对当前记录进行读写操作；一个记录中可以包括几个数据项，也可以只包括一个数据项；数据项可以是字符串数据或数字，也可以是由它们混合组成的记录。图11-1是一个记录文件中的记录组织格式的示意图。

图11-1中的记录文件共有6个记录，记录中可放字符串或数字；每个记录有一个记录号；图右侧表示隐含的指针位置，BEGIN指向第一个记录，END指向记录文件末尾，SAME指向当前记录，NEXT表示为当前记录的下一个记录。

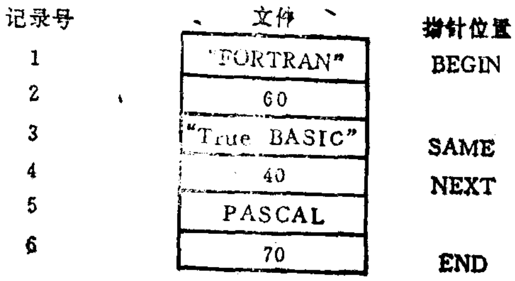

图 11-1

3.记录文件的操作步骤

与顺序文件操作步骤有些类似，但对记录的操作是完全不同的。

(1)记录文件在操作前也要首先使用OPEN语句打开，并指出文件性质及建立方式。

(2)对于新建立的记录文件，要用SET #n RECSIZE〈长度〉语句规定记录长度。

(3)用WRITE #n 语句写入一个记录。

(4)用READ #n 语句读出一个记录。

(5)不使用文件时，用CLOSE #n 语句关闭文件。

二、记录文件的建立

1.记录文件的建立——OPEN语句

与顺序文件的建立相似，使用一个记录文件，不管该文件是否已经存在，均要用OPEN语句建立为当前的操作文件。

OPEN #〈通道号〉：name "文件名"，create〈建立模式〉，organization record

不同的是在记录文件打开时，不用指定访问模式，因为记录文件打开后，既可以向文件写入，也可以由文件读出，组织方式指定为record方式。

例如：

OPEN #1：name "Giu"，organization record，create new

该语句为文件"Giu"打开通道号1，该文件为记录文件，并是新建立的。

若是新建立文件，在open语句中还可指明记录长度，如：

OPEN #1：name "Gju"，organization record，create new，recsize 20

语句中recsize20指定文件中记录长度为20字节。

记录长度设置也可用专用的设置语句，其格式为：

SET #〈通道号〉：recsize〈记录长度〉

其中通道号要与打开文件的通道号相同，〈记录长度〉单位为字节，是一个数字或数字表达式，其最大值限制为2^{24}（大约16兆）。

注意：一个已建立的老记录文件，若其中已写入记录时，则不允许重新设置记录长度。

2.向记录文件写入记录（WRITE#n语句）

记录文件在用OPEN语句打开，并规定了记录长度之后，便可向其写入数据。向记录文件写入数据（记录）修改数据（记录）和追加数据（记录），都要使用WRITE#n语句，其格式为：

(1)WRITE#n：〈表达式表〉

其中，每个表达式可以是数值表达式或字符串表达式，表达式之间以逗号分隔。而且每个表达式对应一条记录，可以把字符串和数值混合写入记录文件。

(2)MAT WRITE #n：〈数组1〉，〈数组2〉，……

每个数组元素在记录文件中各占一条记录。

向记录文件写数据时，是从文件指针所指当前记录开始顺序写入的，每写一个数据，指针向下移动一条。

例7 试建立图11-1所示的记录文件"Giu"。

```
!
OPEN #1: NAME"GJU", ORGANIZATION RECORD, CREATE NEW
SET #1: RECSIZE 20
DO WHILE MORE DATA
   READ A$, Q
   WRITE #1: A$, Q
LOOP
DATA FORTRAN, 60, True BASIC, 40, PASCAL, 70
END
```

本程序中文件的记录长度为20字节。由READ语句读数据赋给变量A $及Q；由 WRITE #n语句将数据写入文件，每次写入两个记录。执行本程序的结果就在磁盘上存放了一个名为Giu、TRU的随机文件。文件共有6个记录（即DATA语句所包含的全部数据项）。

注意，随机文件的内容不可直接在屏幕上显示，只能用读语句将其内容读入内存后，才能向屏幕或打印机输出。

三、从记录文件读取记录（READ #n语句）

存放在记录文件中的记录信息既可以顺序读取，又可以随机读取。

程序通过读语句（READ#n语句）来读取记录中的数据项，并将这些数据项的值分别赋给对应的变量，其格式为：

(1)READ#n：〈变量名表〉

其中，#n应与OPEN #n语句中的通道号相同，变量名表为一个或多个变量名，变量名之间用逗号分开。变量名的类型应与对应读出记录中的数据类型相一致，即写入记录时变量为数字时，读出变量名也应为数字型。

(2)MAT READ #n：〈数组1〉，〈数组2〉，……

这种格式用于读取整个数组，每个数组元素读取一条记录内容。应当注意，数组元素应与记录内容的数据类型相匹配。

例8 在例7中曾建立了一个记录文件“Giu”，下面程序是从记录文件中读取各条记录中的数据，并且有随机读取的功能。

程序及运行结果为：

```
!
OPEN #1: NAME "Giu"
OPEN #2: PRINTER
    INPUT PROMPT "A=": A
    IF A=1 THEN
        SET #1: POINTER BEGIN
        LET I=0
        DO WHILE MORE #1
            LET I=I+1
            READ #1: TEM$
            PRINT #2: USING$"<# # #": I) " ": "TEM="; TEM$
            LET I=I+1
            READ #1: Q
            PRINT #2: USING$"<# # #": I) ; " "; "QUANTITY="; Q
        LOOP
    ELSE
    INPUT PROMPT"RECNUM=": N
    SET #1: RECORD N
    IF N=0 THEN STOP
    IF MOD (N, 2) < >0 THEN
        READ #1: TEM $
        PRINT #2: USING$"RECNUM: # # #":N) , "TEM="; TEM $
    ELSE
```

```
          READ #1: Q
          PRINT #2: USING$ ("RECNUM: # # #": N) , "QUANTOTY="; Q
          END IF
      END IF
  END
  OK.RUN
      1  TEM=FORTRAN
      2  QUANTITY=60
      3  TEM=TRUE BASIC
      4  QUANTITY=40
      5  TEM=PASCAL
      6  QUANTITY=70
  RECNUM: 3     TEM=TRUE BASIC
```

程序解释：执行程序，如果输入1，即令A=1时，打印输出记录文件“Giu”的全部记录内容。否则，输入3给N，则打印第三号记录。

四、记录文件增加记录

对记录文件增加记录有下面两种情况：

1.如果记录文件已存在，则增加新记录。这时必须把文件指针移到文件末尾。它是通过下面语句。

SET#n: POINTER END

来实现的，然后再追加新记录。

2.如果记录文件不存在，则新建一个记录文件，这时文件指针指向文件的开头，也是文件末尾，然后增加记录。

注意：每追加一条记录，文件指针就自动下跳一条记录。

例9 对例7所建立的记录文件“Giu”追加两条记录：“FOXBASE”，90。并用打印机输出整个记录文件的内容。

程序及运行结果为：

```
!
OPEN #1: NAME "Giu"
OPEN #2: PRINTER
SET #1: POINTER END
DO
    INPUT PROMPT "TEM=": TEM$
    IF TEM$="END" OR TEM$="end" THEN EXIT DO
    INPUT PROMPT "QUANTITY=": Q
    WRITE #1: TEM $, Q
LOOP
SET #1: POINTER BEGIN
DO WHILE MORE #1
   READ #1: TEM$, Q
   PRINT #2: TEM$
   PRINT #2: Q
LOOP
```

```
END
OK.RUN
FORTRAN
  60
TRUE BASIC
  40
PASCAL
  70
FOXBASE
  90
```

五、记录文件修改记录

对记录文件的一条或多条记录的修改是通过WRITE语句来实现的。所谓修改记录就是对记录中的内容进行重写。这时应当指出对第n条记录进行修改，使文件指针指向这条记录，才能进行修改。

例10 对例9追加两条记录之后的记录文件“Giu”的第七号记录进行修改，将原记录中的数据“FOXBASE”修改为“ORACLE”，并将第八条记录内容90改为60。然后把整个记录文件的内容由打印机输出。

程序及运行结果为：

```
!
OPEN #1: NAME "Giu"
OPEN #2: PRINTER
DO
   INPUT PROMPT "RECNUM=": N
   IF N=0 THEN EXIT DO
   SET #1: RECORD N
   IF MOD (N, 2) <>0 THEN
      INPUT PROMPT "NEW TEM=": TEM $
      WRITE #1: TEM $
   ELSE
      INPUT PROMPT "NEW QUANTITY=": Q
      WRITE #1: Q
   END IF
LOOP
SET #1: POINTER BEGIN
DO WHILE MORE #1
   READ #1: TEM $, Q
   PRINT #2: TEM $
   PRINT #2: Q
LOOP
END
OK.RUN
FORTRAN
60
```

```
TRUE -BASIC
40
PASCAL
70
ORACLE
60
```

六、多数据项记录文件

在很多实际事务管理中，记录文件的每条记录只有一个数据项是不够的，一般情况下，一条记录里要存放多个数据项，例如，职工登记表中，可能有编号、姓名、性别、民族、年龄、职称、工资等许多数据项，如果每条记录只能存放一个数据项，这样会把一个职工的有关信息存放在多个不同的记录中，对处理和加工职工的有关信息会带来很多方便，甚至会出现许多问题和困难，首先，这样排列极不清晰；其次，由于记录长度固定，而实际数据的长短各不相同，这样必然浪费大量存储空间。

True BASIC可以解决这个问题，多个数据项完全可以合并存放到一条记录中去。解决的方法是：

(1)将多个数据项全部转换成字符串；

(2)把这些字符串加在一起组成一个大字符串；

(3)把大字符串作一条记录存入记录文件中；

(4)把读出的记录分解还原成多个数据项。

例11 建立一个职工情况记录文件，其记录由个人姓名、籍贯、年龄、出生年份和工资组成。

程序及运行结果为：

```
!
DEF pack $ (name$, jig$, age, chu, wage)
   LET pack $=name $ & jig $ & num $ (age) & num $ (chu) & num$ (wage)
END DEF
SUB unpack (s)$, na$, ji$, ag, ch, wa)
    LET na $=s $[1: 20]
    LET ji$=s $[21: 40]
    LET ag=num (s $[41: 48])
    LET ch=num (s $[49: 56])
    LET wa=num (s $[57: 64])
END SUB
OPEN #2: PRINTER
OPEN#1: name"ZGQK", organization record, create newold
ERASE #1
SET  #1: recsize 64
DO WHILE MORE DATA
    READ name $, jig $, age, chu, wage
    WRITE #1: pack $ (name $, jig$, age, chu, waeg)
LOOP
RESET #1: begin
```

```
DO MHILE MORE #1
   READ #1: s $
   CALL unpack (s $, name $, jig$, age, chu, wage)
   PRINT #2: name $; jig$; age: wage
LOOP
DATA "Wang ping       ", "Bejing         ", 45    , 1945, 148.50
DATA "Ying zhenhuai   ", "Shanghai       ", 38    , 1952, 115.60
DATA "Zbo huazhang    ", "Wuhan          ", 50    , 1940, 250.40
DATA "Zhang wei       ", "Nanjing        ", 56    , 1934, 305.00
DATA "Jiang nan       ", "Hangzho        ", 27    , 1963, 86.70
END
OK.run
Wang ping       Bejing          45    148.5
Ying zhenhuai   Shanghai        38    115.6
Zbo huazhang    Wuhan           50    250.4
Zhang wei       Nanjing         56    305
Jiang nan       Hangzho         27     86.7
```

程序中定义了一个pack $函数，用于将一个记录的各数据项压缩为一个字符串，然后写入记录文件的一个记录中。

子程序unpack的作用是将压缩的字符串按各项所需占用的字节长度加以分解。这里姓名和职称分别用20个字符，各占20个字节，其他均按8个字符，各占8个字节。压缩时用num $函数将数据转换成字符串，而在分解时又用num函数将字符串返回数字的形式。

程序执行时是用OPEN# 语句建立一个文件名为ZGQK（职工情况）的记录文件。

语句ERASE# 是为了能够多次执行这个程序，而每次都应将文件内容删除掉。因为下一个语句为规定记录长度（Set# 1：RECSIZE64）语句，这一语句只有在文件内容为空时才合法。当然，如果打开的文件ZGQK是磁盘固有的老文件时，也可将ERASE和SET RECSLZE两语句从程序中除去。而改为指针设置语句，数据从指针处写入。

文件内容是用READ语句从DATA语句中读入内存，每读一个职工信息，组成一个记录后由WRITE# 语句写入，这个语句执行时是调用自定义函数把压缩后的数据项作为一个记录写入文件。

程序的后半部分是将文件中的记录用READ # 语句读入内存（赋值给字符串变量S $）。文件有多少个记录，就读多少次。每读一个记录就调用一次子程序Unpack，将记录的内容分解为五个数据项并分别存入相应的变量。PRINT语句将所要显示的内容输出到显示器上。

§11-4 字节文件

字节文件是由字节的集合构成的文件，它是字节流。

一、字节文件的特点

(1)字节文件是由若干个字节构成的文件。

(2)存储在磁盘上的任何文件都可作为字节文件来处理和加工。例如，BASIC、FORT-

RAN、PASCAL等高级语言的程序文件；True BASIC的程序文件、顺序文件、记录文件；dBASEⅢ的程序文件、数据文件等等。

(3)字节文件总是以字符串形式读入或写出的。

(4)每次可读写任意字节数的字符串。

(5)文件指针可以移动到文件的任意位置上。

字节文件是非常有用的文件，许多高级语言并没有这类文件。

二、字节文件的写入

只有字符串表达式中的字节能被写入字节文件，字符串中每字节占有8位（即一个字节），文件也是由文件指针顺序写入文件中，每写一个字节，指针向下移动一个字节。字节文件是通过写语句写入数据的。它的格式为：

WRITE# n:〈字符串表达式表〉

三、字节文件的读取

由字节文件读取时，只能赋给字符串变量。在读取前要求告诉X字符串的长度（即包含字节数），可以使用SET RECSIZE语句建立读出字串长度，也可以在READ语句中单独定义字节长度，它们的格式为：

SET #n: RECSIZE〈数字表达式〉

READ #n:〈字串变量表〉

或 READ #n, BYTES〈数字表达式〉:〈字串变量表〉

前者先定义读出的记录长度，再用READ语句读出到字符串变量中，后者在READ语句中定义读取的字节长度，从字节文件中每次读入规定的字节数到给定的字符串时，当正确的字节数被读出后，或者文件已达到末尾，则读语句自动停止。

如果在读语句执行中，遇到文件结束了，则产生一个错误（文件结束）。

下面给出一个程序，它可以复制任何类型的文件。输入的文件既可以是二进制码的可执行文件，也可以是文本文件。程序中每次由输入文件读入2k字节到Buffer $中，然后再写入"copyfile"文件中，产生的文件是以TRU为扩展名的，可在DOS中重新命名。程序为：

```
LINE INPUT promot "Enter filemame: ": filename$
OPEN #1: name filename$, access input, organization byte
OPEN #2: name "b: copyfile", access output, organization byte, create newold
ERASE #2
DO
    WHEN EXECEPTION IN
        READ #1, bytes 2048, buffer$
    USE
        LET endflag=1
    END WHEN
    WRITE #2: buffer$
LOOP UNTIL endflag=1
END
```

习 题 十 一

1. 什么是文件，文件中包括哪几种信息？

2. 三种（顺序、记录、字节）方式有何区别？在数据文件处理上有何特点？

3. 指针是什么？怎样确定指针的位置？怎样改变指针的位置？

4. 文件操作语句INPUT#和PRINT#与READ#和WRITE#语句有什么不同？

5. 用顺序方式，编辑一个查找ABC.TXT文件中王林工资的程序，要求显示和打印王林工资额。

6. 用记录方式，编辑一个程序，把ABC.TXT文件变成不可显示的随机文件，找出第五个记录，并显示和打印出该记录的内容。

7. 试用字节方式，改变一个程序名称。

第十二章　图　形

计算机图形处理是个内容十分丰富的课题，应用十分广泛。例如，可用计算机辅助设计（CAD）绘制各种机械图、建筑图、气象图、统计图表等等。本章将介绍True BASIC提供的基本作图功能，并结合管理工作中信息处理的需要，给出一些统计图表的画法。

§12-1　作图环境

True BASIC显示图形使用的是图形窗口，图形窗口不同于编辑窗口和背景窗口，它是以象素为单位控制屏幕，而后者是字符屏，以字符为单位划分屏幕行列数。

一、图形窗口的建立

在True BASIC中为了确定屏幕上一个点的位置，只需在屏幕平面上建立坐标系。在建立坐标系后，一个点的位置便由点的坐标来唯一确定。用户不必关心该点对应的实际象素位置。

当对图形窗口的坐标系不加规定时，系统会自动设置其坐标原点在屏幕的左下角，x轴是0～1，y轴也是0～1。屏幕的左下角、右下角、右上角、左上角的坐标分别是：(0,0)、(1,0)、(1,1)、(0,1)，屏幕坐标图见图12-1所示。

系统自动设置的图形窗口坐标，对具体的作图问题不一定方便，这时可由用户自行设置图形窗口，并由SET WINDOW语句实现。

设置窗口语句格式为：

```
SET WINDOW Xmin, Xmax,Ymin, Ymax
```

其中，四个参数分别是横坐标x和纵坐标y的上下限。

例如，语句

```
SET WINDOW-1, 1, -1, 1
```

在屏幕上建立的坐标系是：坐标原点在屏幕中央，x，y坐标的上下限均为-1，1。见图12-2所示。

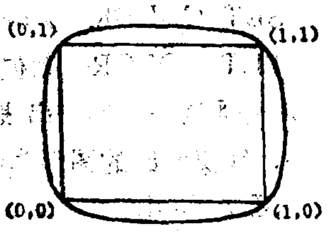

图　12-1

又如，语句

```
SET WINDOW 1900, 2000, 25, 120
```

所建的窗口坐标系见图12-3所示。

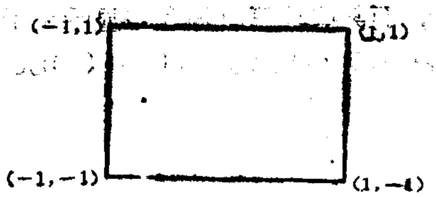

图　12-2

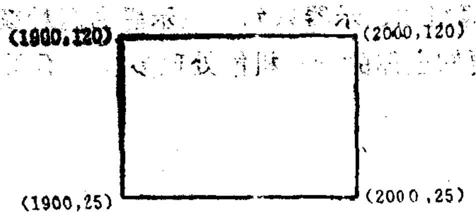

图　12-3

由此可见，图形窗口的坐标系完全可以由用户按自己的实际需要来建立，坐标原点不一定在屏幕上，横、纵坐标的上、下限不一定相同，纵横坐标的单位比例也不一定相同。而且因屏幕是长方形，长宽不同，一般建立窗口坐标时应使横轴的比例大一些，纵横的比例稍短一些，否则会将圆图形画成椭圆图形。

二、设置图形颜色

当不设置颜色时，系统自动按屏幕当前显示的颜色处理，即系统初启时的颜色：背景为黑色，前景为白色(或呈现绿色)。对于带有彩色显示器的用户，可以用SET COLOR语句和SET BACK语句，选择前景和背景的颜色，使图形清晰，并瑰丽多彩。

True BASIC 可以使用各种不同的颜色，甚至还可以套色，这就为我们的使用提供了很大的方便和灵活性。

True BASIC具有十种颜色：

red	红	cyan	青蓝
yellow	黄	brown	棕
magenta	洋红	white	白
green	绿	black	黑
blue	蓝	background	背景

1.设置前景颜色

前景颜色是指所显示的字符或所画的图形的颜色。

True BASIC使用SET COLOR语句设置或改变前景颜色，其格式为：

SET COLOR color $

或　SET COLOR〈数值表达式〉

其中，Color$是一个字符串表达式，它表示颜色名称；而数值表达式，它表示颜色的代码。

例如，欲使输出的图形为红色，则应写作：

SET COLOR "red"

或　SET COLOR 2

这里要注意，当你用颜色名称时，必须用双引号括起来，而用代码就不需双引号了。

正常图形可利用的前景颜色见表12-1所示。

在IBM PC机的正常图形方式下，如使用CGA卡时，前景可同时显示三种颜色，并有两组颜色可供选择。注意，同时使用的只能是属于同一组内的三种颜色，每种颜色均有其代码，以供选择。

某些计算机和显示器只允许显示黑白两种颜色，而某些计算机却允许显示若干种不同颜色和深度，要知道你的计算机能处理多少种不同的颜色，可使用ASK MAX COLOR语句，

表12-1　前景颜色一览表

第一组		第二组	
代码	色　名	代码	色　名
1	green (绿)	4	cyan (深蓝)
2	red (红)	5	magenta (洋红)
3	yellow (黄)	6	white (白)

其格式为：

ASK MAX COLOR Var

其中，Var是一种变量，个此语句将在你的计算机上同时能显示的最大颜色种类数赋给Var，不包括后景颜色。

2.设置后景颜色

后景颜色是指没有任何显示时的屏幕颜色。

True BASIC在选用后景颜色方面就更为灵活了，你可以在不改变前景颜色的条件下使用八种不同的后景颜色，见表12-2所示。

表　12-2

代　码	色　名	代　码	色　名
0	black (黑)	1	blue (蓝)
2	green (绿)	3	cyan (深蓝)
4	red (红)	5	magenta (洋红)
6	yellow (黄)	7	white (白)

设置背景颜色语句格式为：

SET BACK color$

或　　SET BACK〈数值表达式〉

其中，color $和数值表达式分别代表颜色名和颜色代码。无论用哪一种格式效果都是一样的。

你可以在一条语句中同时给出前景颜色和后景颜色，首先写前景颜色，然后划一条斜线，然后再写后景颜色。

例如，语句

SET COLOR "blue/white"

执行这条语句，则前景颜色为蓝色，后景颜色是白色。

三、清除图形窗口

在True BASIC中，使用CLEAR语句清除当前图形窗口，以便画一幅新图，其格式为：

CLEAR

§12-2 标准作图语句

任何复杂的图形总可以看成是由一些基本图形组合而成的。掌握了基本图形的画法、即可由此组合出丰富多姿的各种图形。

一、画点语句

语句格式为：

PLOT POINTS：x_1，y_1；x_2，y_2；……

其中，x_1，y_1是一个点的坐标，x_2，y_2是另一个点的坐标，两点间用分号（“；”）隔开，一个点的两个坐标值之间用逗号（“，”）隔开。执行此语句时，在所给坐标位置上显示一个亮点。注意，所给点坐标必须在当前设置的图形窗口之内，否则显示不出来。

一个PLOT POINTS语句中，可以给出若干对点的坐标，因此在屏幕上画出若干个与坐标相应的点。

画点语句的简化格式是：PLOT x，y，其中x，y是一个点的坐标。执行这个简化格式的画点语句，一次只能画出一个点。

例1 以随机数为坐标，画任意多点。

```
! plot random points like stars
DO
   LET x=rnd
   LET y=rnd
   PLOT x, y
LOOP UNTIL KEY INPUT
END
```

这个程序中没有设置窗口坐标，这样就隐含x轴是0～1，y轴是0～1。执行这个程序时，就在屏幕上随机地显示无数亮点，恰似晴朗的夜空群星闪烁。在Loop语句中加入until key input条件，所以按任一键则结束运行。

二、画线语句

语句格式为：

PLOT LINES：x1，y2；x2，y2；………

也可以使用简略格式为：

PLOT x1，y1；x2，y2……

功能是将以（x1，y1），（x2，y2），…为坐标的点，顺序以直线相连构成折线，当只有两个点的坐标时，则画出一条直线。

如，PLOT LINES：0，0；0，1；1，1；1，0；0，0将绘出一个正方形，顶点坐标分别为（0，0），（0，1），（1，0），（1，1）。

这个语句也可以写作：

```
PLOT LINES: 0, 0; 0, 1;
PLOT LINSE: 1, 1; 1, 0; 0, 0
```

或者写作：

```
PLOT LINES: 0, 0;
PLOT LINES: 0, 1;
PLOT LINES: 1, 1;
PLOT LINES: 1, 0;
PLOT LINES: 0, 0;
```

即只要PLOT LINES语句点的坐标后面是分号（“；”），就会与下一个点连成线段，不论下一点是出现在本语句中，还是在下一个PLOT LINES语句中。

例2，应用画线语句在屏幕上绘出一个红五星。

```
SET MODE: "egahires"
SET WINDOW -5, 5, -4, 4
OPTION ANGLE degrees
FOR X=0 TO 360 STEP 72
 SET COLOR 2
PLOT LINES: 3*cos (X), 3*sin (X) ; cos (X+36) , sin (X+36) ;
NEXT X
END
```

程序开头加了一个OPTION语句，格式是：OPTION ANGLE EDGREES,可以转换为按角度来计算三角函数。

如果要画二条曲线不相连，比如画两个五角星，可以在两个循环之间加入一个PLOT语句，即可使两条曲线之间断开。

如果要对封闭曲线围成的图形着色，即画成一个实心的图形，只要使用PLOT AREA x1，y1，x2，y2：…xn，yn语句即可。

如果将例2中的PLOT LINES：3＊cos(x),3＊sin(x)：cos(x+36),sin(x+36)：改写为PLOT AREA：3＊cos(x);3,sin(x);cps(x+33),sin(x+36);即可画出实心的红五星。

例3 这段程序是轮换采用绿、黄、红色，绘制以随机点为顶点的实心三角形，即三角形的位置和形状均是随机的。

```
SET MODE "egahires"
LET i=1
DO
    SET COLOR mod  (i, 3) +1
    PLOT AREA : rnd, rnd; rnd, rnd; rnd, rnd
    LET i=i+1
LOOP UNTIL KEY INPUT
END
```

例4 绘制利萨如图，见图12-4所示。

程序及运行结果为：

```
! lissajous experiment
OPTION ANGLE  degrees
SET WINDOW -1, 1, -1, 1
DO
INPUT x, y
CLEAR
```

```
FOR i=0 TO 360 STEP 360/100
PLOT sin(x*i), cos(y*i)
NEXT i
PLOT
PAUSE 2
CLEAR
LOOP UNTIL KEY INPUT
END
```

运行此程序，先让你输入两个数值，然后画出一张图，每次画完之后暂停2秒钟，让你观赏一下，然后让你输入两个新值，再画一张新图，这样反复循环，直到你按任一键为止。当输入的数值X为1，y为5时，画出的图形见图12-4所示。

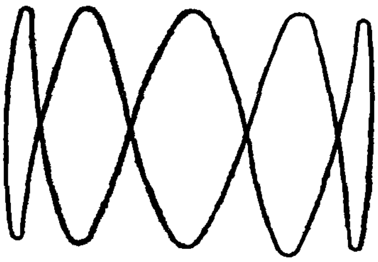

图 12-4

三、在图形中设置文字

为了使图形含义清晰，一目了然，经常在图形中加入一些文字说明。这一要求True BASIC可以允许用PLOT TEXT语句在你的图形所需之处加上字符说明，该语句格式为：

PLOT TEXT, AT x, y: "文字说明"

其中，x，y是附加文字说明的起点坐标，"文字说明"为要显示的文字字符串。

例5 已知某厂去年和今年各季度的产值，分别画两条曲线表示两年产值增长情况。

以x轴表示各季度，设其为0～10，Y轴表示产值，根据数据的分布情况，设为10～30。用两种颜色分别画出两年的产值曲线，并对每条曲线注明它的年份。

```
DIM y (4)
SET MODE "egahires"
SET WINDOW -1, 10, 9, 30
PLOT LINES: 0, 30; 0, 10; 10, 10
FOR i=1 TO 2
    SET COLOR i
    IF i=1 THEN LET t$="1988" ELSE LET t$="1989"
    FOR j=1 TO 4
        READ y (j)
        PLOT LINES: j, y (j) ;
    NEXT j
    PLOT
    PLOT TEXT, AT 5, y (4) : t$
NEXT i
DATA 13, 15.5, 19, 27
DATA 14.9, 18, 28, 26
END
```

例6 绘制直方图，并注明每个直方图条的高度。见图12-5所示。

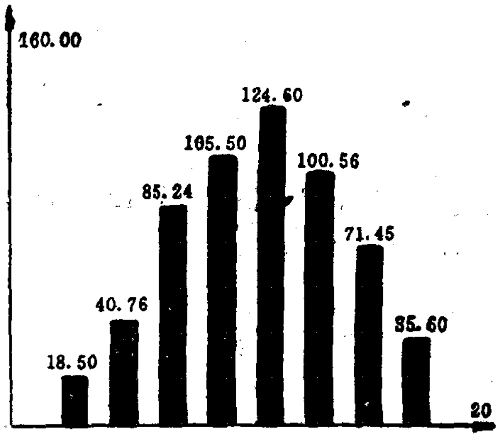

图 12-5

程序及运行结果为：

```
SET WINDOW 0, 20, 0,160
FQR x=2 TO 16 STEP 2
READ v
BOX ARER x,x+1,0,v
PLOT TEXT,at x-1,v+10:USING$("###.##",v)
NEXT x
PLOT TEXT,at0,156:USING $("###.##",160)
PLOT 0,160;0,0;20,0
PLOT TEXT,at 18,1:USING $("###.##",20)
DATA 18.5,40.76,85.24,105.50,124.60,100.56,71.45,35.60
END
```

四、画矩形和多边形

1.画矩形语句

True BASIC用BOX语句可以画各种矩形，而且执行速度非常快。

BOX语句格式为：

BOX LINES left，right，bottom，top其中，left，right，bottom，top分别为所绘矩形的左、右、上、下各边在坐标系中的位置。

若要画成实心的矩形，利用BOX AREA语句，格式为：

BOX AREA：left，right，bottom，top

其中，Left，right.bottom，top含意与BOX LINES说句相同。

例7 用这段程序可以画6个大小各异的矩形和6个实心矩形。

```
SET MQDE"egahires"
SET WINDOW 0,15,0,12
FOR i=1 TO 6
    SET COLOR mod (i,3)+1
    BOX LINES i,i*3/2,i,i*2
NEXT i
FOR i=1 TO 6
    SET COLOR mod(i,3)+2
    BOX AREA i,i*2,i,i*3/2
NEXT i
END
SET MODE "egahires"
SET WINDOW -14,14,-10,10
FOR i=1 TO 10
     SET COLOR mod (i,3)+1
     BOX CIRCLE -i,i,-i,i
NEXT i
END
```

2.画多边形语句

True BASIC画空心多边形是用PLOT语句来实现的，它的一般格式为：

PLOT $x_1, y_1;\ x_2, y_1;\ \cdots;\ x_n, y_n;\ x_1, y_1$

这个语句的功能是光束从第一点（x_1，y_1）开始，画一条线到第二点（x_2，y_2），然后从第二点（x_2，y_2）画一条线到第三点（x_3，y_3），……这样依此类推，最后它画一条线从最后一点（x_n，y_n）到第一点（x_1，y_1），把区域封闭起来。

True BASIC画实心多边形是用PLOT AREA语句实现的，它的一般格式为：

PLOT AREA：x_1，y_1；x_2，y_2；…；x_n，y_n

这个语句的功能是光束从第一点（x_1，y_1）开始，画一条线到第二点（x_2，y_2），然后从第二点（x_2，y_2）画一条线到第三点（x_3，y_3），依次类推，最后自动画一条线从最后一点（x_n，y_n）到第一点（x_1，y_1）把区域封闭起来，并且用当前颜色填满整个区域。

例8 运行下列程序将画出一个亭子，见图12-6所示。

```
SET WINDOW-20,20,-2,32
BOX ARER -1.5,1.5,0,6
plot 0,30;-8,20;-4,20;-12,10;-8,10;-8,0;8,0;8,10;12,10;4,20;8,20;0,30
PROT +4,20;4,20
PLOT -8;10;8,10
END
```

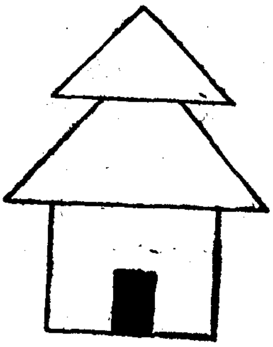

图 12-6

五、画圆和椭圆语句

1.画圆语句

画圆的语句格式为：

BOX CIRCLE left，right，bottom，top

这个语句在参数所决定的正方形框内，绘出以正方形的中心为圆心，以正方形边长之半为半径的圆。

若要画出实心圆，则在上述语句后面加入一个FLOOD语句，其格式为：

FLOOD x，y

语句中的x，y是圆中任一坐标。

例9 试运行下面程序，它将画出五个同心圆，见图12-7所示。

```
SET WINDOW -15,15,-10,10
 FOR i=10 TO 2 STEP -2
 BOX CIRELE  -i,i,-i,i
NEXT i
END
```

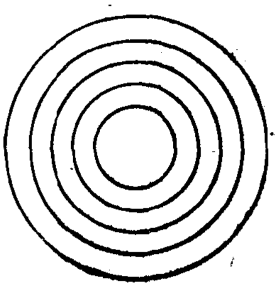

图 12-7

若在BOX CIRCLE -i，i，-i，i后面加一句FLOOD 0，0则可画出实心圆，但只能最内层的一个圆变成实心。

2.画椭圆语句

由BOX CIRCLE语句画圆时，选择left，right，bottom，top值，使它们构成一个矩形框，则在其中所画的就是椭圆。

例如，语句

```
SET WINDOW -2,2,-1,1
BOX CIRCLE -2,2,-1,1
```

画出长轴为4，短轴为2的椭圆。

True BASIC中使用BOX ELLIPSE语句也可以画椭圆，其格式为：

BOX ELLIPSE left, right, bottom, top

其中，left，right，bottom，top的含意与前相同。

在True BASIC中没有实心椭圆的语句，同样，你可采用FLOOD语句来填充。

例10 运行以下程序画出三个同心圆，且在中央画出一个实心椭圆，见图12-8所示。

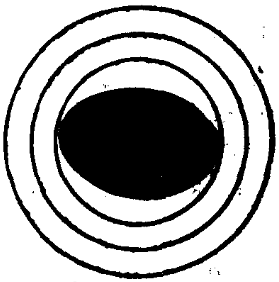

图 12-8

```
SET WINDOW -20,20,-13,13
FOR i=12 TO 8 STEP -2
BOX CIRCIE -i,i,-i,i
NEXT i
BOX ELLIPCE -7,7,-5,5
flood 0,0
END
```

§12-3 动画与图画

一、动画

True BASIC具有实现动画的功能。所谓动画就是在屏幕上模拟物体的运动，这是最有趣的。

当一个图形绘好后，用存图，抹去和再现三个语句，就可以实现图形的运动。

存图语句格式为：

BOX KEEP left, right, bottom, top IN v $

其中，left，right，bottom，top是能将所绘图形包含在内的矩形框的边界。V $是字符串变量名。

其功能是将矩形框内的图形连同前景背景颜色，保存在字符串变量V$中。

抹去语句格式为：

BOX CLEAR left, right, bottom, top

其功能是将left，right，bottom，top围成的矩形框内的图形抹去。使用这个语句时，在不影响其它图形的前提下，应适当地把矩形框加大一些，否则一些边缘上的点可能抹不掉，会遗留一些痕迹。

再现语句格式为：

BOX SHOW v$ AT left, bottom

其功能是在以left，bottom为左下角坐标，将存于v$中的矩形框内的图形，再现在图形窗口内。

顺序使用以上三个语句：首先将所绘图形存储，抹去该图形，再在新位置上再现，再抹

去，又选新位置再现。如此反复执行，就实现了图形的运动。

例11 下面一个程序是显示一个球在屏幕中上下跳动，试运行观赏之。

```
SET WINDOW 0,30,0,20
LET delta=1
LET x=15
LET y=10
BOX CIRCLE x,x+1,y,y+1
FLOOD x+.5,y+.5
BOX KEEP x,x+1,y,y+1 IN ball$
DO
   BOX CLEAR x,x+2,y,y+2
   LET y=y+delta
   IF y<1 or y>19 THEN LET delta =-delta
   BOX SHOW ball$ AT x,y
LOOP UNTIL KEY INPUT
END
```

例12 下面一个程序是显示满天飞舞的红五星。

```
SET MODE "egahires"
SET WINDOW -10,50,-10,40
OPTION ANGLE DEGREES
FOR x=0 TO 360 STEP 72
   SET COLOR 2
   PLOT LINES:3*cos(x),3*sin(x);cos(x+36),sin(x+36);
NEXT x
FLOOD 0,0
LET c=1
LET a=0
LET b=0
BOX KEEP a-3,a+3,b-3,b+3 IN star$
DO
   BOX CLEAR a-3.5,a+3.5,b-3.5,b+3.5
   LET b=b+rnd
   LET a=a+rnd
   IF b<-5 or b>35 THEN LET b=0
   IF a<-5 or a>45 THEN LET a=0
   BOX SHOW star$AT a-3,b-3
LOOP UNTIL KEY INPUT
END
```

例13 试设计一个动画程序，使红、绿、黄三个气球在空中左右飘动。

要确定好这三个气球的相对位和记录这三个气球的矩形位置，使气球左右飘动是利用条件语句的结果，试运行此程序，分析并观赏之。

```
SET WINDOW 0,200,—50,50
SET BACK 1
```

```
LET del=2
LET x=5
LET y=25
SET color 3
BOX CIRCIE x—2,x+2,y—2,y+2
FIOOD x,y
SET COLOOE 2
BOX CIRCLE x+10,x+14,y—2,y+2
FLOOD x+12,y
SET COLOR 1
BOX CIRC le x+4,x+8,y—12,y—8
FLOOD x+6,y—10
BOX KEEP x—2,x+14,y—12,y+2 IN ball$
DO
BOX CLEAR x—2,x+15,y—12,y+3
LET x=x+del
IF x<1 or x>182 THEN LET del=—del
BOX SHOW BALL $ ST x—2,y—12
LOOP S UNTIL KEY INPUT
END
```

二、图画

True BASIC提供一种专用的图画子程序结构。可以实现图形的平移、放大、缩小、变形等多种图形运算。

1.图画的定义

图画子程序的定义格式为：

```
  PICTURE pic (参数, 参数, …)
  …
  PLOT…
  …
END PICTURE
```

其中，PICTURE是定义图画子程序的开始语句，pic是用户定义的图画名字，名字应遵守变量名相同的限制。PLOT语句是产生输出的。END PICTURE是定义图画的结束语句。

图画可以是内部的或外部的，各种参数的传送方式、局变量的可使用性等都类似于内部或外部子程序。

注意：(1)在图画定义中不可执行SET WINDOW或OPEN SCREEN语句。(2)在图画定义中，可以包含EXIT PICTURE语句。当被执行时，控制直接转向调用程序。

2.图画的调用

一般的子程序是由CALL语句来调用，而图画子程序是由DRAW语句来调用。调用图画子程序语句的格式为：

```
DRAW pic (参数, 参数, …) [WITH tran*tran*…]
```

其中，pic是在图画子程序中定义的，WITH是可选项，TRAN表示变换，若有多个变换，其间用星号“*”分隔。

True BASIC中共有五种可能的变换：

SHIFT (a, b) 表示将图画中的所有点 (x, y) 移到 (x+a, y+b) ，从而实现图画的位移。

SCALE (a) 表示将图画中的所有点 (x, y) 变为 (x*a, y*a) ，从而实现将图画放大a倍。

SCALE (a, b) 表示将图画中的所有点 (x, y) 变为(x*a, y*b)，即按横向放大a倍，纵向放大b倍来改变图形的大小，改变图形的横纵比例。

ROTATE (a) 表示将图画绕窗口坐标原点逆时针方向将图画旋转a角(弧度或角度)。

SHEAR (a) 表示将图画中所有垂直线按顺时针方向倾斜a角 (弧度或角度) 。

例14 把一个三色套环见图12-9所示，定义为图画shan，主程序调用这个图画。

```
picture shan
   SET COLOR 1
   BOX CIRCLE —7,3,—3,7
   SET COLOR 2
   BOX CIRCLE —3,7,—3,7
   SET COLOR 3
   BOX CIRCLE —5,5,—8,2
END PICTNRE
SET WINDOW —20,20,—13,13
DRAW SHAN
END
```

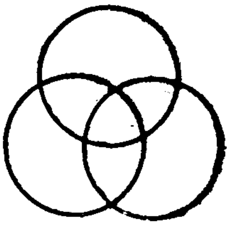

图 12-9

此程序含有一个名为shan的无参数图画定义和主程序。象子程序定义一样，当程序运行时，图画的定义被绕过，你可在主程序的前面或后面，甚至可以在主程序中间定义这个图画，程序在执行时决不会自动“陷进去”，仅当执行DRAW语句时，shan这个图画才被执行。

例15 由程序画一座有四扇窗和一个门的房子，窗是由画窗子程序完成的。主程序三次调用画房子子程序，每次附加不同的变换，结果在屏幕上画出三座位置，形状各异的房子。

程序及运行结果见图12-10所示。

```
! draws three houses using picture
SET WINDOW 0, 12,0,12
SET BACK  "green"
OPTEION ANGLE DEGREES
DRAW HOUSE With scale(1)*shift(7,8)
DRAW HOUSE WITH scale(1.5)*shift(1.8,6)
DRAW HOUSE WITH ROTATE(15)*scale(2,1.6
*SHIFT(6.2, 2.8)
EED
PICTUIE HOUSE
   SET COLOR "cyan"
   PLOT AREA:—1,—1;—1, 1;1, 1;1, 1;1, —1
   SET COLOR 5
   PLOT AREa:—1, 1;0, 1.5;1, 1
  PIOT AREA—1, 1;—1, 1.4;—3, 1.4;—3, 1
```

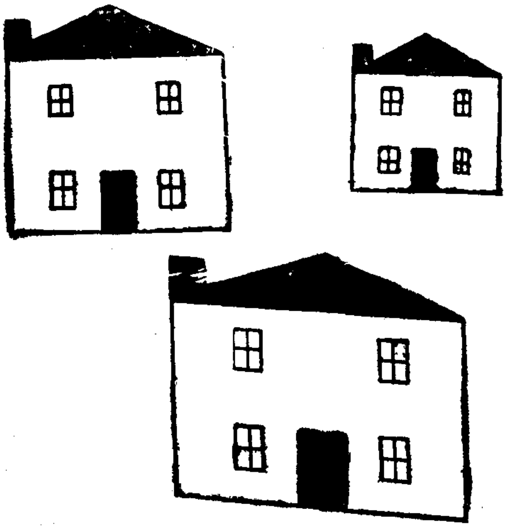

图 12-10

```
    SET COLOR 6
    OIPT AREA:—.15,—1;—,15,—3;15,—3;15,—1
    DRAW WINDOW WITH SHIFT(—.5, —.5)
    DRAW WINDOW WITH SHIF(—.5, 5)
    DRAW WINDOW HTIN SHIF(.5, .5)
    DRAN WINDOW HTIN SHIF(.5, —.5)
    END PICTURE
PICTURE WINDOW
    SET COLOR "background"
    PLOT AREA:—.1, —.2;—1., 2;.1,.21,=.2
    SET COLOR 6
    PLOT —.1, 0;1, 0
    GLET 0, —.2;0.2
END PICTUR
```

§12-4　图形输入与多窗口操作

一、图形输入

所谓图形输入是利用True BASIC提供的语句，可由键盘或鼠标器实现图形象素的输入。True BASIC提供了GET POINT和GET MOUSE两个语句用于图形输入。

1.GET POINT语句，其格式为：

GET POINT：x，y

其中，x和y是窗口坐标中的一个点坐标。其功能是由用户区键盘或鼠标器确定屏幕上的一个点。执行语句时，一个图形十字光标显示在当前点上按一下回车键或鼠标器后，光标当前位置的点坐标值被赋给x，y。

当用户系统中没有鼠标器装置时，用键盘上的光标键移动光标，用回车输入当前十字光标位置的点。

例16　画红五星。程序为：

```
GET POINT, y1
PLOT x1, y1
GET POINT x2, y2
PLOT x2, y2,
GET POINT:x3, y3
PLOT x3, y3
GET POINT:x4, y4
PLOT x4, y4
GET POINT:x5, y5
PLOT x5, y5
SET COLOR 2
PLOT AREA:x1'y1;x5, y5;x3, y3;x2, y2;x4, y4
END
```

2.GET MOUSE语句

GET MOUSE语句是GET POINT语句补充形式，是为了使用鼠标器设计的，能否使

用此语句决定于计算机系统中是否配备有关设备。其语句格式为：

```
GET MOUSE: x, y, state
```

其中，x，y是当前光标中心上的窗口坐标值，state反映出鼠标器当前的状态和变量，它的取值及状态含义，见表12-3所示。

表12-3　鼠标器状态代码

取　值	状　　态
0	鼠标器键未按下
1	鼠标器正被拖着走
2	在此点键被按下
3	在此点键被释放

有关鼠标器的使用详细情况，可参考有关资料。

二、多窗口操作

True BASIC允许将一个屏幕分成几个窗口，可在不同窗口中显示文字或绘制图形。同一时间内，只能对一个窗口进行操作（即使用一个窗口进行输出或绘图），称为当前窗口。True BASIC系统在一般情况下，是使用#0通道号代表整个屏幕的。在程序设计中，可以将屏幕划分为几个不同部分，每部分可设置为一个窗口。

1.窗口的打开

打开一个新窗口，使用open # expr语句给新窗口确定一个通道号（expr），open语句为窗口所作的事类似于一个文件的操作。

打开一个窗口的语句格式为：

```
OPEN #n: SCREEN left, right, bottom, top
```

其中，left，right，bottom，top四个数字分别指出打开窗口的四个顶点在屏幕上的坐标位置，屏幕的水平、垂直方向坐标范围均为0～1，打开的新窗口坐标不能超出这一范围的限制。

如要在屏幕左下角建立一个新窗口，见图12-11所示，则使用语句

```
open #1: Screen 0, 0.4, 0, 0.6
```

在执行此语句后，所有的后继 PRINT，INPUT，PLOT和DRAW语句都将在这一窗口中进行。

打开新的窗口可以与原先打开的窗口部分重叠，这时新窗口将部分复盖原来打开的窗口。

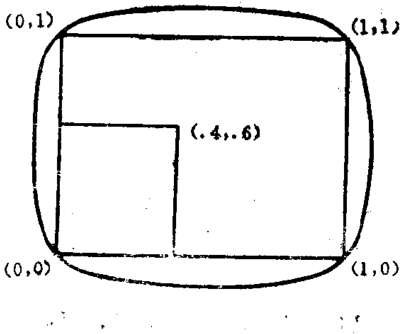

图　12-11

2.窗口的变换

当在程序中打开了几个窗口时，当前工作窗口是最后打开的一个窗口，如要转换到先前打开的窗口中工作，可以使用Window语句实现窗口的变换。变换窗口的语句格式为：

```
WINDOW #n
```

执行此语句后，当前工作窗口变换到#n所指定的窗口。以后的全部屏幕操作在此窗口内

进行，True BASIC还会记住每个窗口的当前颜色，窗口坐标等等参数，并执行Window语句时，自动进行参数变换。可以使用ASK语句查询当前窗口的坐标及其它参数。

3.在窗口内设置光标

如已打开并使用窗口时，Set CURSOR语句总是在当前窗口内设置光标，而不再是在整个屏幕中定位光标。光标位置仍以当前窗口中的行列值给出。

如　SET CURSOR 2, 3

是在当前窗口中由顶部向下第二行第三列处设置光标。注意不要将光标设置到当前窗口之外，否则，会得到错误信息。可用ASK MAX CURSOR语句查询当前窗口光标的限制。

4.关闭窗口

当打开的窗口不用时，使用CLOSE #n 语句可以关闭指定窗口，与关闭一个文件通道一样，关闭的通道号又可任意使用了。

关闭一个窗口后，True BASIC自动转换到#0窗口，如要改为其它窗口，仍要用window语句。

例17　请分析下面程序，且运行结果见图12-12所示。

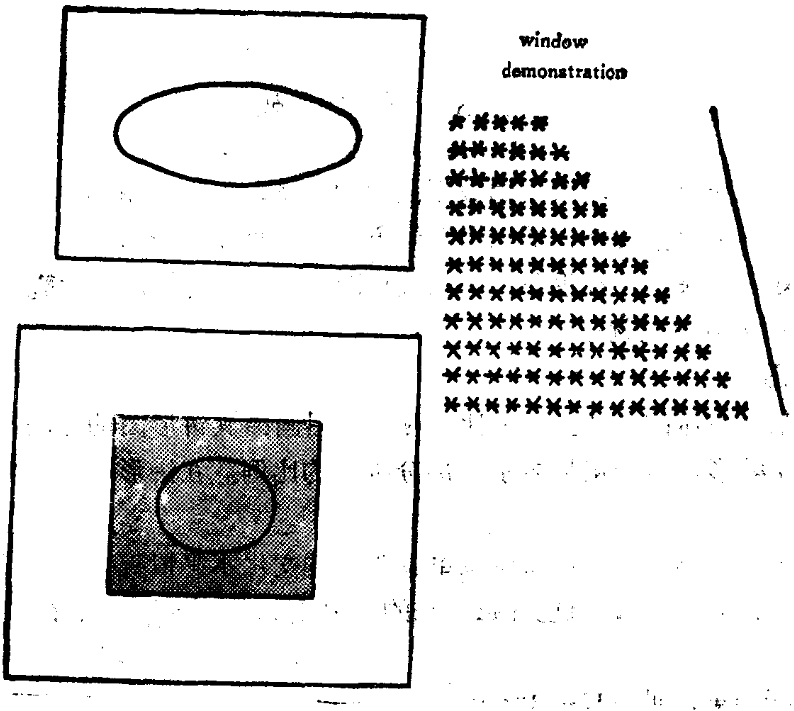

图　12-12

```
OPEN #1: screen 0, 49, 0, 0.5
SET WINDOW -10, 10, -10, 10
BOX LINES -10, 10, -10, 10
BOX AREA -5, 5, -5, 5
OPEN #2: screen .5, 1, 0, 1
SET CURSOR 2, 5
PRINT "window"
PRINT "demonstration"
```

```
SET CURSOR 5, 1
FOR i=5 TO 15
   FOR j=1 TO i
      PRINT "*" ;
   NEXT j
   PRINT
NEXT i
OPEN#3:screen 0, 0.49, 0.6, 1
OET WINDOW —8, 8, —8, 8
BOX LINES —7, 7, —7, 7
BOX CIRCLE —5, 5, —3, 3
WINDOW #1
SET COLCR 2
BOX CIRCLE —3, 3, —3, 3
WINDOW #2
SET WINDOW  —12, 12, —12, 12
PLOT 10, —6;4, 8
END
```

§12-5 统计图的绘制

在信息处理工作中，常涉及大量数据之间的关系，如果仅用列表方式平罗列数值，往往不易看出数据之间的对照关系，而采用图示方法则较直观，一目了然。所以，统计图是经常用来表现信息处理结果的一种手段。常见统计图有曲线图、饼形图、直方图等，因篇幅限制仅介绍饼形图和直方图的例子。

一、饼形图

当要研究某一整体的各部分所占比例时，常用画出其饼形图的方法来表示。将图形按比例分割成若干个扇形，每一扇形表示一部分所占的比例。在饼形图中，各部分的大小与比例情况一目了然。

例18 画一张饼形图，用以表示某单位职工的文化水平情况，设文化水平按学历分为：大专、高中、初中、小学四档，且已知每一档的人数占总人数的百分比依次为10.5%，40.5%，35%，14%。

程序及运行结果，见图12-13所示。

```
DIM  e(10) , f(10)
INPUT PROMPT "n=" :n
FOR i=1 TO n
   INPUT PROMPT  "the data" :e (i)
   LET f (i) =e (i) *360/100
NEXT i
SET WINDOW —2, 2, —1.5, 1.5
OPTION ANGLE DEGREES
BOX CIRCLE —1, 1, —1,1
```

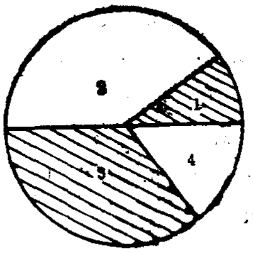

图 12-13

```
LET s=0
FOR i=1 TO n
   IF N=4 THEN
      SET COLOR mod (i, 2) +1
   ELSE
      SET COLOR mod (i, 3) +1
   END IF
   PRINT i;e (i) ; "%"
   LET sl=sl+f (i)
   FOR r=s TO sl—1
      PLOT AREA:0, 0;cos (r) , sin (r) ;cos (r+1) , sin (r+1)
   NEXT r
   LET s=s+f (i)
NEXT i
END
OK.RUN
   1  10.5%
   2  40.5%
   3  35%
   4  14%
```

运行程序时，当出现提示“n=”时，回答4，在提示“the data”后，输入：10.5，40.5，35，14等四个数据项。运行输出不同颜色区分的大小扇形组成的饼形图，并在图的左侧以相应的颜色依次标出各扇形区代表的百分数。注意n不要超过6，否则不清晰。

例19 绘制有同期对比的竖式直方图。设运行时对“n”的回答是4。输入的第一批数据是：150，170，140，180，第二批数据是：160，130，190，140。注意n不要超出5。

程序如下，运行结果见图12-14所示。

```
DIM e (5) , d (5)
INPUT PROMPT "n=" :n
MAT INPUT e (n) , d (n)
SET WINDOW —8, 15, 90, 200
PLOT LINES:0, 100;14, 100
PLOT LINES:0, 100;0, 190
FOR i=1 TO n
   SET COLOR 3
   PRINT i;c (i)
   SET COLOR 2
   PRINT "…" ;i;d (i)
   SET COLOR 3
   PLOT AREA:2*i—1, 100;2*i—1,e (i) ;2*i,e (i) ;2*i,100
   SET COLOR 2
   PLOT AREA:2*i—0.5, 100;2*i—0.5, d (i) ;2*i+0.5, d (i) ;2*i+0.5, 100
NEXT i
END
OK.RUN
```

1	150
…1	160
2	170
…2	130
3	140
…3	190
4	180
…4	140

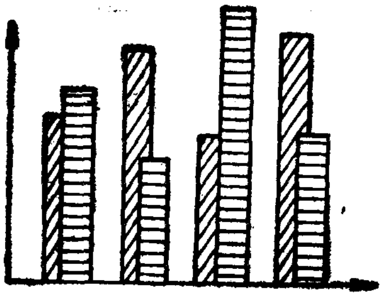

图 12-14

习 题 十 二

1.画正方形内切一个圆。

2.试画一面五星红旗。

3.设计一个小球的动画程序，该小球可按给定正弦曲线运动。

4.设计一个程序，可对指定区域中的图形进行放大、旋转等操作。

5.利用设置窗口操作，将屏幕划分为两个窗口（左右各半），使其能在一个窗口绘图，在另一个窗口中显示文字。

附　录

附录一　ASCII 码表

CHAR	DEC	OCT	CHAR	DEC	OCT	CHAR	DEC	OCT	CHAR	DEC	OCT
NUL	0	0	SP	32	40	@	64	100	\	96	140
SOH	1	1	!	33	41	A	65	101	a	97	141
STX	2	2	"	34	42	B	66	102	b	98	142
ETX	3	3	#	35	43	C	67	103	c	99	143
EOT	4	4	$	36	44	D	68	104	d	100	144
ENQ	5	5	%	37	45	E	69	105	e	101	145
ACK	6	6	&	38	46	F	70	106	f	102	146
BEL	7	7	'	39	47	G	71	107	g	103	147
BS	8	10	(	40	50	H	72	110	h	104	150
HT	9	11	)	41	51	I	73	111	i	105	151
LF	10	12	*	42	52	J	74	112	j	106	152
VT	11	13	+	43	53	K	75	113	k	107	153
FF	12	14	,	44	54	L	76	114	l	108	154
CR	13	15	—	45	55	M	77	115	m	109	155
SO	14	16	.	46	56	N	78	116	n	110	156
SI	15	17	/	47	57	O	79	117	o	111	157
DLE	19	20	0	48	60	P	80	120	p	112	160
DC1	17	21	1	49	61	Q	81	121	q	113	161
DC2	18	22	2	50	62	R	82	122	r	114	162
DC3	16	23	3	51	63	S	83	123	s	115	163
DC4	20	24	4	52	64	T	84	124	t	116	164
NAK	21	25	5	53	65	U	85	125	u	117	165
SYN	22	26	6	54	66	V	86	126	v	118	166
ETB	23	27	7	55	67	W	87	127	w	119	167
CAN	24	30	8	56	70	X	88	130	x	120	170
EM	25	31	9	57	71	Y	89	131	y	121	171
SUB	26	32	:	58	72	Z	90	132	z	122	172
ESC	27	33	;	59	73	[	91	133	{	123	173
FS	28	34	<	60	74	\	92	134	|	124	174
GS	29	35	=	61	75	]	93	135	}	125	175
RS	30	36	>	62	76	∧	94	136	~	126	176
US	31	37	?	63	77	—	95	137	DEL	127	177

附录二　True BASIC语法总结

在下面的语法描述中，用var代表任意串或数字变量，expr代表任意串或数字表达式，省略号（…）表示该语句中的项可以多于或少于所列出的项。例如：

PRINT expr1, expr2, …

表示PRINT语句中可以包含任何数量的表达式，甚至没有表达式。

一、简单语句

ASK MARGIN var

ASK ZONEWIDTH var

DATA item1, item2, …

END

GET KEY var　(var 必须是数字变量)

INPUT var1, var2, …

INPUT PROMPT expr $: var1, var2, …

LINE INPUT var1$, var2$…

LINE INPUT PROMPT expr$ var1$, var2

LET var=expr

LET var1, var2, …=expr

LET var$[i, j]=expr

PAUSE expr expr(必须是数字)

PLAY music $ (music$可以是串常量,变量或表达式,并且只能有音乐字符和空格)。

PRINT expr1, expr2, …

PRINT expr1, expr2, …

PRINT expr1, expr2; …

PRINT USING format$: , expr1, expr2, …

(format$可以是串常量，变量或表达式，并且必须包括格式字段和字母字符)。

READ var1, var2, …

RESTORE

SET MARGIN expr

SET ZONEWIDTH expr(expr必须是数字表达式)

STOP

SOUND freq, time

(freq必须是数字变量，它给出以赫兹为单位的频率数；time必须是数字变量，它给出以秒为单位的持续时间)。

二、循环控制结构

在下面的语法描述中，var必须是数字变量，first，last和size可以是任意数字表达式；condition可以是任何逻辑表达式；语句块可以包含任意数目的语句或结构。

```
DO
...
...} block of statements
...
LOOP
DO WHILE condition
...
...} block of statements
...
LOOP
DO
...
...} block of statements
...
LOOP UNTIL condition
DO WHILE condition
...
...} block of statements
...
LOOP UNTIL condition
```

WHILE条件或UNTIL条件，可以在DO或LOOP语句中出现，或者在两者中都出现或只在其中一个中出现，或者都不出现。

```
EXIT DO
EXIT FOR
FOR var=first TO last STEP size
...
...} block of statements
...
NEXT var
```

三、判断结构

在下面的语法描述中，condition可以是任意逻辑表达式；statement可以是任意一行语句blocks可以包含任意数量的语句或结构，甚至无语句或结构。

```
IF condition THEN statement
IF condition THEN
...
...} block of statements
...
END IF
IF condition THEN statement1 ELSE statement2
IF condition THEN
...
...} block1
...
ELSE
```

```
…
…} block2
…
END IF
IF condition1 THEN
…
…} block 1
…
ELSEIF condition2 THEN
…
…} block 2
…
ELSEIF condition3 THEN
…
…} block 3
…
ELSE
…
…} block n
…
END IF
```

你可以使用任意多个ELSEIF块，以及零个或一个ELSE块。

```
SELECT CASE espr
CASE test1, test2, …
…
…} block 1
…
CASE test3, tost4, ··
…
…} block 2
…
CASE ELSE
…
…} block 3
…
END SELECT
```

你可以使用任意多个CASE块，每个CASE语句可以有任意个检测。这些检测可以有以下列形式：

```
constant
low TO high
IS operator constant
```

constant，low和high必须是与SELECT CASE语句中expr相同类型（数字或串）的常量。operator必须是一个关系操作符。

四、MAT语句

```
MAT array1=array2
MAT array1, array2, …=array
MAT array1, array2, …=expr
MAT array1=array2+array3
MAT array1=array2*array3
MAT array1=constant*array2
MAT INPUT array1, array2, …
MAT INPUT array(? )
MAT INPUT PROMPT expr $: array1, array2,…
MAT LINE INPUT: array1$, array2$,…
MAT LINE INPUT PROMPT expr$: array1$, array2$,…
MAT PRINT array1, array2, …
MAT PRINT array1; array2; …
MAT PRINT USING format$: array1$; array2$, …
MAT READ array1, array2, …
```

五、程序部件

```
CALL subr(explr, expr2, …)
CHAIN expr$
CHAIN expr$ WITH(arg $)
CHAIN expr$, RETURN
CHAIN expr$  WITH(arg), RETURN
DECLARE DEF func1, func2, …
DEF func(var1, var2, …)=expx
DEF func(var1, var2, …)
  …
  …
LET func=expr
  …
  …
END DEF
EXIT DEF
EXIT SUB
LIBRARY  "filename1" , "filename2" , …
PROGRAM name
PROGRAM name(var $)
SUB subr(var1, var2, …)
  …
  …
END SUB
```

有些把程序部件称为一个程序单位，等等，都是一个含义。

六、图

下面语句中，所有坐标系（x，y等等）都是图形窗口坐标系。

```
ASK BACK var
ASK BACK var$
ASK BACKGROUND COLOR var
ASK BACKGROUND COLOR var$
ASK COLOR var
ASK COLOR var$
ASK MAX COLOR var
ASK CURSOR var$
ASK CURSOR x, y(x, y为字符屏幕坐标)
ASK MAX CURSOR x, y(x, y为字符屏幕坐标)
(以下语句中 left, right, bottom和top是图形屏幕坐标。)
ASK SCREEN left, right, bottom, top
ASK WINDOW left, right, bttom, top
BOX AREA left, right, bottom, top
BOX CIRCLE left, right, bottom, top
BOX CLEAR left, right, bottom, top
BOX ELLIPSE left, right, bottom, top
BOX KEEP left, right, bottom, top IN var$
BOX LINES left, right, bottom, top
BOX SHOW var$ AT left, bottom
BOX SHOW var$ AT left, bottom USING constant$
BOX SHOW var$ AT left, bottom USING expr
(constant $可以是“and”，“or” “xor”，expr或必须是数字)
CLEAR
CLOSE# expr (expr必须是数字)
DRAW pic
DRAW pic WITH trans
DRAW pic WITH trans*srans*…
FLOOD x, y
GET MOUSE: x, y state
GET POINT: x, y
MAT PLOT AREA: array
MAT PLOT LINES: array
MAT PLOT POINTS: array(array必须是数组，而且有确定的大小)
OPEN# expr: SCREEN left, rigpt, hottom, top
(togt,rigqzt,bottom,toq,是屏幕坐标系中的坐标值)
PICTURE pic(var1, var2, …)
```

…
…} block of statemens
…

END PICTURE （block中可以不包含SET WINDOW或OPEN SCREN语句）
PLOT x, y
PLOT x, y;
PLOT x1, y1; x2, y2,…;
PLOT AREA: x1, y1; x2, y2,…;
PLOT LINES: x1, y1; x2, y2,…;
PLOT POINTS: x1, y1; x2, y2,…;
PLOT TEXT, AT x, y: expr$
SET BACK expr
SET BACK expr$
SET BACKGROUND COLOR expr
SET COLOR expr
SET COLOR expr$
SET CURSOR expr$ （expr$可以是“off”或“on”）
SET CURSOR x, y
SET WINDOW left, right, bottom, top
（这4个表达式必须是数字，它们确定了窗口坐标系的左、右、底和顶。）
WINDOW #expr

七、文件处理

下面的语句可以和任何文件类型一起使用（文本，记录，字节）#expr表示一个“#”号和数字表达式的标记，它们代表一个在1—999之间的通道号。

ASK #expr: ACCESS var$
ASK #expr: FILESIZE var
ASK #expr: MARGIN var
ASK #expr: NAME var$
ASK #expr: ORGANIZATION var$
ASK #expr: POINTER var$
ASK #expr: RECORD var
ACK #expr: RECSIZE var
ASK #expr: ZONEWIDTH var

（MARGIN或ZONEWIDTH仅适用于正文文件；RECRD和RECSIZE仅适用于记录文件。）

CLOSE #expr
ERASE #expr
INPUT #expr: var1, var2,…
LINE INPUT #expr: var1$, var2$,…
MAT LINE INPUT #expr var1$, var2$,…

```
OPEN # expr: NAME  expr$, ACCESS accmode$,
                          CREATE crtmode$,
                          ORGANIZATION orgmode$
                          RECSIZE recsize
```

ACCESS, CREATE, ORGAZIATION和RECSIZE是任选项，NAME是必须的。各种可能的串是：

选择项	串	意思
ACCESS	input	只 读
	output	只 写
	outin	读/写
CREATE	new	建立新文件
	old	使用已存在的文件
	newold	使用现存文件或建立新的
ORGANIZATION	text	正文文件
	record	记录文件
	byte	字节文件

(recsize必须是数字。)

```
OPEN #expr: PRINTER
PRINT #expr: expr1, expr2; …
PRINT #expr; expr1: expr2…
PRINT #expr, USING expr$ expr1, expr2, …
```

(# expr: 必须是正文文件通道)

```
READ #expr; varlv, ar2, …
MAT READ #expr: array1, array2, …
```

(#expr必须是个记录或字节文件通道。)

```
READ #expr, BYTES expr: var1$, var2$, …
```

(#expr必须是个字节文件通道。)

```
RESET #expr: BEGIN
RESET #expr: END
RESET #expr: NEXT
RESET #expr: RECORD expr
RESET #expr: SAME
```

(NEXT, RECORD expr, 和SAME仅适于记录文件。)

```
SET #expr: MARGIN expr
SET # expr: POINTER BEGIN
SET # expr: POINTER END
SET # expr: POINTER NEXT
SET # expr: POINTER SAME
SET #expr： RECORD expr
SET #expr： RECSIZE expr
SET #expr： ZONEWIDTH expr
```

(MEXT，POINTER，RECORD和RECSIZE仅能和记录文件共用)。

(#expr必须是记录号或文件通道号)。

附录三　从BASIC到True BASIC的转换软件

一、概述

大多数BASICA用户在改用true BASIC时，可能已有一些BASICA应用程序。如果要将这些程序转换到true BASIC，用手工逐条修改，工作量将是很大的。true BASIC 2.03版提供了一个自动转换软件，可以自动完成由BASICA到true BASIC语言的转换工作，并对不能转换的部分作出标记，使你很容易找到，并作出手工修改。

转换软件包的内容有：

CONVERT·EXE　　转换程序

DEFLIB·TRU　　被转换程序使用的库

转换软件可在DOS系统中运行（不需要进入True BASIC系统）。转换软件本身不需要DEFLIB·TRU文件，但该文件必须与被转换的程序在同一软盘上。

转换软件将文本形式BASICA源程序转换为True BASIC代码。过程是逐行读取BASICA文件，作必要的语法修改，再重写到输出文件上，两种BASIC语言中相同的部分不经修改，只将TRUEBASIC中没，有的BASICA语句加以修改，或加“＊＊＊”标记，以待手工完成。还有一些BASICA语句被转换成调用特殊的子程序，这些子程序包含在DEFLIB·TRU库文件中。

二、转换软件的启动

使用转换软件的环境是，具有192k以上内存的IBM PC机或IBM PC兼容机。一个软盘驱动器及2.00以上版本的DOS。

1.建立文本格式BASIC文件

首先要有一个文本格式的BASIC文件。假若文件名为BASFILE·BAS，应首先在BASICA环境中检查行号，并以ASCII码方式存盘，在BASIC启动后使用

```
LOAD "BASFILE·BAS"
```

装入程序，先查看程序第一行的行号是否大于44，且行号间距至少应大于5。如不符合上述要求时，则应先修改行号，以便转换程序在行号10到44之间插入一段序言；行号之间留有间距，是为了能将BASICA的多语句行变成多行语句结构。按上述要求重新编好后，以ASCII文件存盘，可用命令：

```
SAVE "BASFILE·TRU" , A
```

这里建议使用TRU作为扩展名，因为它是True BASIC的缺省扩展名。当然也可以使用其它扩展名。

用SYSTEM命令从BASIC中退出。

2.转换软件的使用

在A驱动器中插入有转换软件的软盘，并输入命令：A>CONVERT

转换程序启动后系统提示：

BASICA 源文件名：or “stop”：BASFILE↙

True BASIC目的文件名：TRUFILE6↙

其中源文件名是要转换的BASICA文件，名为BASFILE。目的文件名为生成的True BASIC文件名，名为TRUFILE，文件名下的横线表示由用户输入。

如被转换的文件扩展名是TRU，可以省略，否则应加扩展名。

转换完成后，屏幕上将显示“ALL DONE”。当提供另外的文件名时，又可转换其它文件。当不再需要转换时，用“stop”作为回答，就可以返回到DOS。

三、手工修改不能转换的内容

BASICA源程序代码中，90％以上的内容均可用转换软件完成转换工作，其余部分均已加“＊＊＊”标记，可以很容易地用手工加以修改。步骤是：

(1)由HELLO启动True BASIC。

(2)装入包含有已转换好的TRUFILE的软盘，并用OLD TRUFILE 调入该文件，再使用 True BASIC屏幕编辑功能修改。输入RUN命令或按F9键运行程序时，编译系统自动检查程序中的语法错误，并将光标停在错误处，给出错误信息。对于转换软件不能转换的内容均标有“＊＊＊”标记，这显然也是一个语法错误，可用手工加以改正。

由于True BASIC编译程序一次只能给出五个错误信息，因此有可能要重复使用RUN命令查找错误。

四、转换软件的具体说明

应该指出，转换软件的目的只是产生一个在功能上和原来的BASICA源程序等价的True BASIC程序，因此相对于True BASIC语言来说，转换生成的程序可能不是最简明或效率最高的。例如：转换软件并没有设法使用True BASIC 中提供的高级特性；另外，对于BASICA中的一些不能在True BASIC中使用的语句，是由所提供的外部子程序或定义的函数来实现的，这些都可能引起效率的降低。

1.插入的序言

转换程序在每一个被转换程序的起始部分（1～44行）都设置一个标准序言。序言中包含一个LIBRARY语句，用以说明程序中要使用库文件DEFLIB·TRU，在True BASIC中不存在的多数BASICA语句将通过调用DEFLIB·TRU中的子程序来转换。True BASIC中没有的一些函数，也由调用DEFLIB·TRU中定义的函数来解决。

因为DEFLIB·TRU中的所有函数，对于被转换程序来说是外部的，所以在序言中还加入一个DECLARE DEF语句，用来说明那些可能由被转换程序调用的函数。最后，序言还包含一个OPTION BASE语句，以设置和BASICA相一致的缺省数组下界。

2.数据的精度

由于True BASIC所有算术运算都使用精度为14位数字的双精度浮点数进行。所以对BASICA中的单精度与双精度的处理是相同的。相应的转换函数（CSNG，CDBL）被省略。BASICA中整型量与实型量之间的转换

```
LET  a%=b!
```

被转换成下述True basic语句

```
LET a=round(b)
```

转换软件也可以准确地处理具有相同名字而类型不同的变量。例如，在BASICA中，x，x!，x%和x#是不同的，转换器照下面的样子分离x并替换（假定x是单精度）：

BASICA	True BASIC
x%	x—i
x!	x—s
x#	x—d

3.布尔表达式

转换软件能正确处理布尔表达式，包括IMP和EQV操作。对于非操作，括号包括了全部子句，由于在BASICA中非操作是允许的，而在True BASIC中则不允许。

如果关系表达式缺乏关系操作符，则转换软件为其加上一个“<>0”。

转换软件无法转换在运算中出现的布尔值表达式，如

X=Y<Z

该式在BASICA中是合法的（X根据Y<Z的真假被赋为0或—1）。但在True BASIC中它是不合法的，被转换为：

LET X=Y * * *<Z

转换程序也不处理逻辑表达式，包括逻辑操作和数字或位逻辑操作。

4.数组

转换程序在引用数组以前需要有一个数组定义语句，True BASIC要求数组在使用前一定要先定义其维数及大小，不允许稳含地定义（BASICA允许对下标由0到10的一维数组可以不定义）。

BASICA允许数值变量与数组变量同名，而True BASIC则不允许。如果变量名和前面定义的数组名相同时，但如果是未定义的数组和变量同名，则变量名不变，由True BASIC编译程序发现这个错误。下面是一些例子。

BASICA	True BASIC
100 DIM a(100)	100 DIW a(100)
200 a(3)=2	200 LET a(3)=2
210 a=4	210 LET a—t=4

假定没有用DIM语句定义a数组

BASICA	Trub BASIC
200 a(3)=2	200 LET a(3)=2
210 a=4	210 LET a=4

如果DIM语句出现在对同名变量赋值语句之后。

BASICA	True BASIC
100 a=4	100 LET a=4
200 DIM a(100)	200 DIM a(100)
210 a(3)=2	210 LET a(3)=2

对于这两种情况，转换程序都不能对变量名作修改，但在运行时True BASIC编译程序会指出错误。通过在程序开头放置所有DIM语句的办法，就可以避免这类错误。

以上对True BASIC转换软件作了简单的介绍，若想了解更详细情况：可参阅有关手册。

附录四　True BASIC与DOS 系统间的接口方法

为了在True BASIC环境中直接执行DOS操作系统的命令，True BASIC(2.03版)提供了接口程序SHELL·TRU，在用户程序中调用SHELL程序（作为库文件），就可在程序中直接执行DOS操作系统的任何命令，得到的结果与在DOS系统中完全一样。

在True BASIC(2.03版)系统盘中有一个SHELL.TRU的程序，该程序中有两个子程序：DOS和SHELL。调用这两个子程序可以使你不必退出True BASIC就可执行 DOS 的命令。

现举例说明调用方式，如在主程序中加入

Library “SHELL，TRU” 及

CALL DOS(“DIR”)语句，执行调用语句时，系统先清屏，后将括号内的参数（字符）传送给DOS的命令处理程序；后者按DOS命令去执行参数中给出的命令，但在命令执行完时，处于等候状态，直到用户的键盘输入任一字符后才回到True BASIC状态。

如果使用的操作系统版本号低于3.0时，应在系统配置文件中加入PATH=drive：，对于3.0以上的DOS操作系统，没有这个必要。

在CALL DOS(A $)中，A $可以是任何操作系统的命令（包括所需参数），如果A $=“ ”，则相当于返回DOS操作系统。可用EXIT命令再返回到True BASIC。

另一调用子程序是SHELL，其功能及调用方法与DOS子程序完全一样，仅仅不清除屏幕。

下面通过一个程序说明SHELL、TRU的使用及进入DOS的方法。

```
! exama-1
! use of shell·tru
LIBRARP “d: \tb\shell·tru”
DO
    PRINT “D>” ;
    LINE INPUT prompt “” : a$
    CALL dos(a$)
LOOP UNTIL a$= “ ”
CLEAR
END
```

```
! exama-2
! use of shell·tru
LIBRARY “d: \tb\shell·tru”
DO
LINE INPUT prompt “: a $
CALL shell(a$)
LOOP UNTIL a$= “  ”
CLEAR
END
```

上述程序可以改成外部子程序库，在任何True BASIC程序调用时可以实现进入DOS环境的操作，执行上述调用的限制是机器内存。如果执行的命令程序太大，内存无法容纳时，则系统出错并返回操作系统。

附录五 True BASIC 与其他可执行程序（·EXE文件）间的接口方法

True BASIC(2.03版)提供了一个与可执行程序（·EXE文件）接口的子程序EXEC·TRC使用这一程序作库文件，可从任何True BASIC程序中调用执行磁盘上的EXE文件，因而可以很方便容易地实现通用程序“工具箱”的联接。这使True BASIC应用程序的开发集成化变得非常容易。

一、EXEC·TRC 外部程序库

EXEC·TRC子程序库包括EXEC(name$, arg$)与EXEC·RETURN (name$, arg$)两个子程序。

1.SUB EXEC(name$, arg$)

EXEC使DOS终止当前程序，并装载名为name$的（扩展名为·EXE或·COM）文件。它类似于CHAIN name$(arg$)。

name$是新程序的文件名，它必须包括扩展名（·EXE或COM），需要时也可包括驱动器名或目录路径。

arg$允许传递数据参数到调用程序。若调用程序是另一个true BASIC汇集文件，且是由PROGRAM name(string$)语句开始的，则它能检查此参数，对其它语言，arg$则象在DOS命令行中所输入的参数一样传送到执行程序中，在arg$为空串时，不传送任何参数。

name$和arg$都必须少于127个字符长度。

调用EXEC以前，系统关闭所有打开的文件。

2.SUB EXEC－RETURN(name $, arg$)

EXEC-RETURN象CHAIN name$(arg$)，RETURN语句一样，将当前程序保留在内存中，当执行完调用程序后，恢复执行当前程序。它有些类似于true BASIC语言系统中的SHELL子程序（见附录六），它更有效，因为命令处理器不必再装入内存，从而节省了内存空间和磁盘的I/O操作。

这里name$是调用新程序的文件名，必须有扩展名（·EXE或·COM).，根据需要可包括驱动器名或目录路径。

arg $是一个可选择参数，它传递一个数据串到调用程序中。若调用新程序是另一个汇集true BASIC文件，应是以PROGRAM name(string$)语句开始，它可以传送并检查此参数。对其它语言，arg$就象在DOS命令行中输入的参数一样传递。当不需要传送参数行时，则可以使用空串作为第二个参数，这时在调用程序中，不需要使用PROGRAM语句。

name和arg$必须少于127个字符。

调用EXEC—RETURN之前，将关闭所有打开文件。注意：不要使用EXEC— RETURN来执行一个“停启和常驻内存”的程序（也称TSR程序、台式附件或上托）。如果这样做了，将引起错误（959），产生提示：“Can′treclaim memory”。这时，TSR程序的内存将被破坏。最安全的作法是重新启动机器。

二、应用EXEC·TRC的方法

下面通过一个例子说明EXEC·TRC的调用方法。

372806

哈尔滨船舶工程学院 图书馆
中文藏书

```
! exama 7-1
LIBRARY "exec.trc"
DO
   LINE INPUT prompt "Enter a name of program: " ; a $
   IF a $="" THEN EXIT DO
   LINE INPUT prompt "Enter a parameter list: " ;b $
   CALL exec-return(a $, b $)
LOOP
END
```

此程序执行时，调入库文件EXEC·TRC，由键盘输入要执行的任何·EXE程序，可以包括驱动器名及路径名，例如：

```
   OK·RUN↙
Enter a name of program: c: \PC TOOLS·EXE↙
Enter a parametr list: ↙
```

则程序由C上目录中读出PC TOOLS·EXE并装入内存执行。以后就象在DOS系统中执行该程序一样。

若在第一次输入时打入回车，则程序自动退出循环，并结束运行。参数输入项可以输入必要的参数表，也可直接回车（参数项为空）。

在程序中通过EXEC·TRC调用执行其它可执行程序时，主要限制是机器内存，当内存容量不足装入所调程序时，系统出错，并提示：

```
Out of memory
```

参考文献

1 高永泉主编.结构化BASIC程序设计教程—True BASIC.北京：电子工业出版社，1990。

2 余静等编著.True BASIC结构程序设计教程.沈阳：东北工学院出版社，1991

3 杨振生等编著.True BASIC语言程序设计教程.合肥：中国科学技术大学出版社，1991

4 John GKemeng.Thomas E kurtz Structured BASIC programming

Document generated by Anna’s Archive around 2023-2024 as part of the DuXiu collection (https://annas-blog.org/duxiu-exclusive.html).

Images have been losslessly embedded. Information about the original file can be found in PDF attachments. Some stats (more in the PDF attachments):

```
{
  "filename": "MTAyMDQ4MTYuemlw",
  "filename_decoded": "10204816.zip",
  "filesize": 15675434,
  "md5": "247d7cde902fffa341b9bddb5a1486f6",
  "header_md5": "4a90498b676bec77f2377a4aff29cdb6",
  "sha1": "a1ca02f8642f95a709fef868e5e2c0ae58067f47",
  "sha256": "758ffd847ca60cff6a52d4fcc7f554449de975e4df4cf20706cace794ebe9c01",
  "crc32": 3276763726,
  "zip_password": "",
  "uncompressed_size": 15853163,
  "pdg_dir_name": "True BA SIC\u2559\u2229\u2564\u2558\u255c\u00df\u2563\u2563\u2557\u00bb\u2502\u2560\u2568\u2265\u2554\u03a6\u255d\u255e_10204816",
  "pdg_main_pages_found": 226,
  "pdg_main_pages_max": 226,
  "total_pages": 233,
  "total_pixels": 1631989656,
  "pdf_generation_missing_pages": false
}
```